1

DB2 update

Special edition

In this issue

A xephon PUBLICATION

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Colin Smith
E-mail: info@xephon.com

Subscriptions and back-issues

A year's subscription to *DB2 Update*, comprising twelve monthly issues, costs $380.00 in the USA and Canada; £255.00 in the UK; £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 2000 issue, are available separately to subscribers for $33.75 (£22.50) 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.

Contributions

When Xephon is given copyright, articles published in *DB2 Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) 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.

SMF exit for DB2

Alongside the normal DB2 applications that run against our production DB2 datasharing environment we have an ever-growing workload connecting via TCP/IP and DDF. The problem we have is that for our traditional applications we wish to collect SMF accounting records, but for the distributed workload we do not. The existing tracing controls within DB2 offer the facility to start a trace for a named plan or, indeed, a list of plans (up to a limit), but not the facility to exclude a named plan from being traced. The sheer volume of accounting records produced by our distributed workload has, on occasion, flooded the SMF buffers with the resulting loss of data not only for DB2 but for other software that writes to SMF. Along with this, the SMF housekeeping and reporting processes are effectively working on large volumes of data – a significant proportion of which is not required.

As mentioned, the ideal solution would be an EXCLUDE option within the DB2 TRACE command because this would stop the records being written at source. The added advantage of this would be saving on CPU overhead caused by collecting and writing the trace record in the first place. At present this is not an option, so the next best thing is to intercept the accounting records before SMF collects them. The following code is an SMF exit that looks for specific PLAN names within DB2 101 accounting records and directs SMF to ignore them. The source code contains a table of PLAN names that we want SMF to ignore. For our distributed applications, the PLAN name is a constant, DISTSERV, and is the only one to feature in our list. Sample JCL is supplied giving details of installing this exit as a z/OS user mod.

The software levels involved are z/OS 1.4 and DB2 V6 and V7, although the code should be valid for most combinations of MVS and DB2.

SOURCE CODE FOR THE SMF EXIT

```
IEFU83 TITLE 'SMF RECORD EXIT'
```

```
*********************************************************************
*   DESCRIPTION
*   THIS ROUTINE IS USED FOR IEFU83/4/5 TO SUPPRESS DB2 RECORD
*   TYPE 1Ø1S FOR SELECTED PLAN NAMES.
*   THERE IS A HARDCODED TABLE OF PLAN NAMES IN THIS CSECT.
*   INTERFACE
*   ENTRY POINT = MODU83
*   ENTRY REGISTERS = REG1 POINTS TO A 4-BYTE ADDRESS OF THE
*                     RECORD DESCRIPTOR WORD (RDW) OF THE SMF
*                     RECORD TO BE PROCESSED BY THE EXIT.
*   RETURN CODES = Ø - WRITE THE SMF RECORD (KEEP)
*                  4 - DO NOT WRITE THE SMF RECORD (DELETE)
*   ATTRIBUTES
*          LOCATION = LPA
*          STATE = SUPERVISOR
*          AMODE = 31
*          RMODE = ANY
*          KEY = KEY Ø
*          MODE = ANY (CAN BE SRB OR XMEM, LOCKED)
*          SERIALIZATION = NONE
*          TYPE = REENTRANT
*          NAME = IEFU83
*          ALIAS = IEFU84, IEFU85
*  !!! WARNING !!!
*  BECAUSE THE EXIT MAY BE INVOKED IN SRB MODE (IEFU84/5) IT CANNOT
*  ISSUE ANY SVCS.
*********************************************************************
         EJECT
MODU83   CSECT
MODU83   AMODE 31
MODU83   RMODE ANY
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12                  BASE REG
R13      EQU   13                  SAVEAREA REG
R14      EQU   14                  RETURN ADDRESS
R15      EQU   15                  ENTRY ADDRESS & TEMPORARY BASE
         USING MODU83,R15
         SAVE  (14,12),,'MODU83 TESCO &SYSDATE &SYSTIME'
         LR    R12,R15             LOAD BASE ADDR
```

```
         DROP  R15
         USING MODU83,R12
         LR    R9,R1               SAVE PARAMETER LIST ADDRESS
         L     R9,Ø(R9)            GET ADDRESS OF SMF RECORD
         USING SM1Ø1,R9
         SLR   R15,R15             SET RC = ACCEPT RECORD
         CLI   SM1Ø1RTY,1Ø1        IS IT TYPE 1Ø1
         BNE   RETURN              NO, LET IT WRITE
*********************************************************************
*  SUPPRESS DB2 TYPE 1Ø1 FOR PLAN 'DISTSERV'
*********************************************************************
         LA    R3,SM1Ø1END         TO START OF SELF-DEFINING SECTION
         USING QWAØ,R3
         LA    R4,Ø(R9)            START OF SMF RECORD (RESET BITØ)
         A     R4,QWAØ1PSO         TO PRODUCT SECTION
         LH    RØ,QWAØ1PSL         LEN OF PRODUCT SECTION
         LR    R3,R4               START OF PRODUCT SECTION
         USING QWHS,R3
         AR    R4,RØ               END OF PRODUCT SECTION
NXTHDR   DS    ØH
         CLI   QWHSTYP,QWHSHCØ2    CORRELATION HEADER ?
         BE    CHKPLAN             YES, CHECK PLAN NAME
         AH    R3,QWHSLEN          TO NEXT HEADER
         CR    R3,R4               ANY MORE HEADERS
         BL    NXTHDR              YES, KEEP LOOKING
         B     RETURN              DO NOT SUPPRESS
*
* CHECK PLAN NAME AGAINST TABLE OF THOSE WE SUPPRESS
*
         USING QWHC,R3
CHKPLAN  DS    ØH
         LA    R1,PLANTAB
CHKNXT   CLI   Ø(R1),C' '          END OF TABLE ?
         BE    RETURN              YES, KEEP SMF RECORD
         CLC   QWHCPLAN,Ø(R1)      PLAN NAME MATCH ?
         LA    R1,L'QWHCPLAN(R1)   TO NEXT ENTRY
         BNE   CHKNXT              NO, KEEP LOOKING
         LA    R15,4               SET RC = SUPPRESS RECORD
*
*  RETURN WITH RC IN R15
*
RETURN   DS    ØH
         L     R14,12(R13)         RESTORE RETURN ADDRESS
         LM    RØ,R12,2Ø(R13)      RESTORE REGS ZERO THRU 12
         BSM   Ø,R14               RESTORE CALLER MODE AND RETURN
*
PLANTAB  DS    ØH
         DC    CL8'DISTSERV'
         DC    CL8' '              SPARES
         DC    CL8' '
```

```
         DC    CL8' '
         DC    CL8' '
         DC    C' '                END-OF-TABLE
         DSNDQWAS SUBTYPE=ALL
         END
```

JOB TO INSTALL SMF EXIT

```
//JOBCARD
//S1       EXEC PGM=GIMSMP,
//         PARM='PROCESS=WAIT',
//         DYNAMNBR=12Ø
//SMPCSI   DD DISP=SHR,DSN=SMPE.ZOS14.GLOBAL.CSI
//SYSLIB   DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=SYS1.MODGEN,DISP=SHR
//         DD DSN=DB2.V71SMP.SDSNMACS,DISP=SHR
//SMPHOLD  DD DUMMY
//SMPCNTL  DD DDNAME=SYSIN
//SYSIN    DD  *
  SET BDY(GLOBAL).
  REJECT  S(TMSMFØ7) BYPASS(APPLYCHECK) .
  RESETRC.
  RECEIVE S(TMSMFØ7) .
  SET BDY(TGT14A) .
  APPLY S(TMSMFØ7) REDO CHECK RC(RECEIVE=Ø).
  APPLY S(TMSMFØ7) REDO RC(APPLY=Ø).
//SMPPTFIN DD  DATA,DLM=ff
++USERMOD(TMSMFØ7) REWORK(2ØØ41381) /* IDENTIFY CHANGE DATE YYYYDDDV */.
++VER(ZØ38) FMID(HBB77Ø7)
   /*************************************************************
  *
  *:ELEMENTS:     SRC - MODU83
  *:DESCRIPTION:  SMF EXIT
  *:FUNCTION:     SMFWTM (SMF RECORD WRITE) EXIT TO SUPPRESS
  *               SELECTED DB2 (TYPE 1Ø1) RECORDS.
  *
   *************************************************************/.
++JCLIN.
//LINK      EXEC PGM=IEWL,PARM='LET,NCAL,REUS,RENT,AC=1'
//SYSLMOD   DD   DSN=SYS1.LPALIB,DISP=SHR
//SYSLIN    DD   *
 INCLUDE SYSPUNCH(MODU83)
 MODE AMODE(31),RMODE(ANY)
 ENTRY  MODU83
 NAME   MODU83(R)
/*
++SRC(MODU83) DISTLIB(TESSRCE) TXLIB(TESSRCE) .
ff
```

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
//*********************************************************
//STEP01Ø  EXEC PGM=IEFBR14
//SORTOUT  DD DSN=&HLQSAM..&JOBNAME..QUIESCE,
//            DISP=(MOD,DELETE),
//            UNIT=WORK,SPACE=(TRK,(1,1))
//*********************************************************
//***      GENERERATE QUIESCE-STATEMENT
//*********************************************************
//STEP02Ø  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
//*******************************************************
//STEP030   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,(1,10),RLSE)
//SYSUT1    DD  DISP=(NEW,PASS),UNIT=WORK,SPACE=(CYL,(1,10),RLSE)
//SYSERR    DD  DISP=(NEW,PASS),UNIT=WORK,SPACE=(CYL,(1,10),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     : 28-5-2003                                               */
/* 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=1001: no parm specified                              */
/*            rc=1002: database not found                             */
/*            rc=1003: unknown DB2 subsystem                          */
/* ****************************************************************** */
Trace o
Arg param
If param='' then do
   erc =1001
   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 ********************** */
```

Loet J Polkamp
Database Administrator (The Netherlands)

How to verify DB2 UDB back-ups

It is all very well taking DB2 back-ups using the BACKUP command (>db2 backup db <*db-alias*> to <*drive*>), but how do you know that the back-up is usable? DB2 UDB (7.2 and 8.1) provides a utility called **db2ckbkp** to perform this task.

This article looks at the db2ckbkp utility. The *DB2 UDB V8 Command Reference* manual (SC09-4828-00) describes the function of this utility quite distinctly, as "… *This utility can be used to test the integrity of a backup image and to determine whether or not the image can be restored*." This is a very important although often overlooked task.

I ran all the commands in this article on a Windows 2000 machine running DB2 8.1 FP1. I used the SAMPLE database as the test database.

To get a list of all possible parameters, use the command:

```
>db2ckbkp -h
```

Let's look at some practical uses of this command. I took a back-up of the SAMPLE database to the C:\backups directory. Which of the possible parameters is the best one to use? If you don't specify any parameters, then what you get back when you invoke the command is a line saying that the image verification was successful (or not).

The utility can be invoked in one of two ways:

- By CDing to the directory containing the back-up image and issuing the command with the file name as a parameter:

```
C:\backups\SAMPLE.0\DB2\NODE0000\CATN0000\20030323>db2ckbkp  185549.001
```

- By issuing the command using the full back-up path name:

```
C:\>db2ckbkp
C:\backups\SAMPLE.0\DB2\NODE0000\CATN0000\20030323\185549.001
```

Note that in this second example you need to specify the full path name even if you have only one back-up file in the directory (you can't just specify the directory name).

Both of the above invocations produce a couple of lines of output, but the key line to look for is the last line, which says "Image Verification Complete - successful.".

What about the other parameters? The **-h** parameter dumps the media header and performs an image verification. It produces the following output:

```
C:\>db2ckbkp -h
C:\backups\SAMPLE.0\DB2\NODE0000\CATN0000\20030323\185549.001

=====================
MEDIA HEADER REACHED:
=====================
        Server Database Name           -- SAMPLE
        Server Database Alias          -- SAMPLE
        Client Database Alias          -- SAMPLE
        Timestamp                      -- 20030323185549
        Database Partition Number      -- 0
        Instance                       -- DB2
        Sequence Number                -- 1
        Release ID                     -- A00
        Database Seed                  -- 3E8E829E
        DB Comment's Codepage (Volume) -- 0
        DB Comment (Volume)            --
        DB Comment's Codepage (System) -- 0
        DB Comment (System)            --
        Authentication Value           -- 255
        Backup Mode                    -- 0
        Backup Type                    -- 0
        Backup Gran.                   -- 0
        Status Flags                   -- 1
        System Cats inc                -- 1
        Catalog Partition Number       -- 0
```

```
DB Codeset                     -- IBM-1252
DB Territory                   --
Backup Buffer Size             -- 4194304
Number of Sessions             -- 1
Platform                       -- 5
```

The proper image path would be:

```
SAMPLE.0\DB2\NODE0000\CATN0000\20030323\185549.001
[1] Buffers processed:  ########

Image Verification Complete - successful.
```

This output might be useful if somebody just sends you the back-up file. You can then re-create the directory structure, because the output gives you the instance, back-up date, database name, and partition number, in fact everything you need to recreate the C:\backups\SAMPLE.0\DB2\NODE0000\CATN0000\20030323 path. You can then recreate the full path name and use the file in a recovery situation.

The **-H** parameter does the same as the **-h** parameter, except that it does not perform an image verification – all it does is display the media header.

If you use any of the **-a**, **-c**, **-d**, **-l**, **-o** parameters, you get back numerous pages of output, with the last line indicating whether the image verification was successful or not. I am not sure how useful the output produced by using these parameters is – it may be required by the IBM support centre if you report a problem with the back-up.

I think that it is vitally important that once I take a DB2 back-up I verify it. I would do this using the **db2ckbkp** command without any parameters. This does not substitute for performing regular disaster recovery tests, but should be part of that process. I hope I have shown how simple the invocation of the **db2ckbkp** command is. If you do not currently verify you back-ups once you have taken them, I would strongly recommend that you do so – it is a quick step that could prove decisive if you ever find yourself in the situation where you need to use your back-ups in a real disaster scenario!

C Leonard
Freelance Consultant (UK)

An ISPF-SQL interface

What the following REXX EXEC does is quite simple: basically it takes an SQL SELECT statement as input, executes it, and fills the result set into an ISPF table for further processing. The name of the ISPF table is given back to the calling program (plus some other information, like errormessage, sqlcode, number of rows returned, etc). I found that this is a much easier approach to reading data from a DB2 table than the usual procedure:

1 Establish DSNREXX environment.

2 Establish DSNREXX connection.

3 Establish DB2 connection.

4 DECLARE CURSOR.

5 PREPARE SQL statement.

6 OPEN CURSOR.

7 FETCH result rows.

8 CLOSE CURSOR.

9 Examine all possible error conditions for steps 1-8.

If you are a DBA or programmer who often needs to write REXX programs that read DB2 data, you will find DSN2ISPF very handy.

There is one thing you need to beware of: if your ISPF table contains one or more columns that are *not* part of the result set from the DB2 table, specify placeholder values in your SQL statement! Otherwise DSN2ISPF won't work properly.

For example, your ISPF table should consist of three columns – COL1, COL2, and COL3. You're about to read COL1 and COL2 from your DB2 table and COL3 is going to be calculated later in your program. Write an SQL SELECT similar to the following one:

```
SELECT DB2_COL1, DB2_COL2, 0
FROM your-table
```

```
WHERE where-condition
```

If you do so, the ISPF table column COL3 will be initially set to zero. Of course you can choose any initial value you want.

Note: DSN2ISPF contains two subroutines, DSN2ISPF_LOCALDB2 and DSN2ISPF_DB2LOC. DSN2ISPF_LOCALDB2 gives you the name of a DB2 subsystem that is running on the System/390 LPAR where DSN2ISPF is currently executing. That DB2 subsystem is used for the DSNREXX connect. DSN2ISPF_DB2LOC gives you the location name of a given DB2 subsystem-identifier. This location name is used for the SQL CONNECT TO statement. At our site, we use these two subroutines as external REXX EXECs that can be used by any other program. This is a very useful standardization method too.

DSN2ISPF uses some input parameters that are documented in the source code itself. I've chosen default values for most of them, which is sufficient for about 99% of all the occasions when you are using DSN2ISPF.

Following the code for DSN2ISPF you will find a tiny sample program to show the usage of DSN2ISPF in its most basic form.

If you have any questions or comments, feel free to contact me at peter.adlersburg@t-systems.at.

DSN2ISPF

```
/* Rexx
   Proc DSN2ISPF
   Do SQL Select Stmt and Create an ISPF Table containing the Data
   Returns ISPF Tablename (DSN2nnnn)
   Who: Peter Adlersburg / T-Systems Austria

   DSN2ISPF VGETs its Parameters from the ISPF-Var-Pool

   1) DSN2SSID      : the DB2 Subsystem-ID (eg DSN)
   2) DSN2STMT      : the SQL Select Statement that will be executed
   3) DSN2VARS      : a list of Variable names that make up the ISPF
                      table (Format var1,var2,...)
   4) DSN2KEYS      : a list of Variable names that make up the Key of
                      the ISPF Table (if DSN2ISPF_KEYS contains no
                      value an unkeyed ISPF Table is created; Format
```

```
                      key1,key2,...)
   5) DSN2TRCE      : Causes a REXX Trace to be activated (normal
                      REXX Trace Values will be used; default 0 (Off))
   6) DSN2ELVL      : Error-Level:
                      Ø ... No Errors and Informations
                      1 ... Errors only (default)
                      2 ... Informations and Errors
   7) DSN2ROWS      : the maximum number of Rows that will be fetched
                      (default is 999)
   8) DSN2ISOL      : Desired Isolation level (CS,UR,RR,RS; UR is the
                      default)
   9) DSN2TABB      : Behaviour if NO-ROWS-FOUND-Condition is met
                      TAB   ... Build empty ISPF Table and return
                      NOTAB ... Do NOT build ISPF Table and return
                                (default)
  1Ø) DSN2ISPP      : Parameters that will be used for ISPF Table
                      Creation. The default is 'NOWRITE,REPLACE'

  11) DSN2LOCL      : Determines whether a Connect to DSN2SSID can
                      be made via another DB2 subsystem or not
                      This can be useful when DDF of DSN2SSID is down
                      during execution
                      1 ... Connect is allowed via another subsystem
                            (default)
                      Ø ... Connect is done via Locationname of
                            of DSN2SSID
   1), 2) and 3) are mandatory parameters !!

   All Parameters are deleted from ISPF Variable Pool after Execution
   of DSN2ISPF

   DSN2ISPF VPUTs some Variables after Execution that can be used by
   the caller

   DSN2RETC ..... ErrorCode
   DSN2ERRM ..... Errormessage
   DSN2RSLT ..... Number of Rows fetched

   DSN2RETC-Values:

        Ø  Successfull Execution
        1  Max.Number of Rows reached  (Trunc.ISPF-Table returned)
        4  No Rows returned            (Empty ISPF-Table returned)
        8  Parameter-Error
        12 ISPF- or DSNREXX-Error
    -nnnnn SQL-Error
*/
SIGNAL ON SYNTAX NAME DSN2ISPF_SYNTAX ;
Address TSO ;
DSN2RETC = '4' ;                        /* Default Returncode       */
```

```
DSN2RSLT = '0' ;                          /* Default Number of Result-*/
                                          /* Rows                     */
DSN2TNAM = 'DSN20000' ;                   /* Default ISPF-Tablename   */
"ISPEXEC VERASE (DSN2RETC,DSN2ERRM,DSN2RSLT) BOTH" ;
/************************ ErrorLevel *********************************/
"ISPEXEC VGET DSN2ELVL" ;
if rc = 8 | DSN2ELVL = '' then,           /* Set Default Error level  */
   DSN2ELVL = '1' ;
if DSN2ELVL < '0' | DSN2ELVL > '2' then do ; /* Check Error level     */
   DSN2ELVL_ERR = DSN2ELVL ;
   DSN2ELVL = '1' ;
   retc = DSN2ISPF_MSG('Invalid Errorlevel: 'DSN2ELVL_ERR,'E') ;
   DSN2RETC = '8' ;
   signal DSN2ISPF_END ;
end ;
retc = DSN2ISPF_MSG('Current Errorlevel: 'DSN2ELVL,'I') ;
/************************ REXX Tracing *******************************/
"ISPEXEC VGET DSN2TRCE" ;
if rc = 8 | DSN2TRCE = '' then,           /* Set Default REXX Tracing */
   DSN2TRCE = 'O' ;
Rx_Cmd = "Trace "DSN2TRCE ;
interpret Rx_Cmd ;                        /* Activate REXX Tracing    */
retc = DSN2ISPF_MSG('Current Tracelevel: 'DSN2TRCE,'I') ;
/************************ Max.Number of Rows *************************/
"ISPEXEC VGET DSN2ROWS" ;
if rc = 8 | DSN2ROWS = '' then,           /* Set Default MaxRows      */
   DSN2ROWS = 999
if datatype(DSN2ROWS) <> 'NUM' then do ;
   retc = DSN2ISPF_MSG('Invalid DSN2ROWS-Value 'DSN2ROWS,'E') ;
   DSN2RETC = '8' ;
   signal DSN2ISPF_END ;
end ;
DSN2ROWS = trunc(DSN2ROWS) ;              /* Truncate DSN2ROWS        */
retc = DSN2ISPF_MSG('Max.Number of Rows set to: 'DSN2ROWS,'I') ;
/************************ Isolationlevel *****************************/
"ISPEXEC VGET DSN2ISOL" ;
if rc = 8 | DSN2ISOL = '' then,           /* Set Default-ISOLevel     */
   DSN2ISOL = 'UR' ;
if DSN2ISOL <> 'UR' &,
   DSN2ISOL <> 'RR' &,
   DSN2ISOL <> 'CS' &,
   DSN2ISOL <> 'RS' then do ;
   retc = DSN2ISPF_MSG('Invalid DSN2ISOL-Value 'DSN2ISOL,'E') ;
   DSN2RETC = '8' ;
   signal DSN2ISPF_END ;
end ;
retc = DSN2ISPF_MSG('DSN2ISOL Value set to: 'DSN2ISOL,'I') ;
/************************ ISPF-Table Behaviour ***********************/
"ISPEXEC VGET DSN2TABB" ;
if rc = 8 | DSN2TABB = '' then,           /* Set Default              */
```

```
   DSN2TABB = 'NOTAB' ;
if DSN2TABB <> 'TAB' &,                    /* Check Values             */
   DSN2TABB <> 'NOTAB' then do ;
   retc = DSN2ISPF_MSG('Invalid DSN2TABB-Value 'DSN2TABB,'E') ;
   DSN2RETC = '8' ;
   signal DSN2ISPF_END ;
end ;
retc = DSN2ISPF_MSG('DSN2TABB-Value set to: 'DSN2TABB,'I') ;
/************************ ISPF Table Parameter ************************/
"ISPEXEC VGET DSN2ISPP" ;
if rc = 8 | DSN2ISPP = '' then,            /* Set Default              */
   DSN2ISPP = 'NOWRITE,REPLACE' ;

retc = DSN2ISPF_MSG('DSN2ISPP-Value set to: 'DSN2ISPP,'I') ;
/************************ DB2 Subsystem ID ****************************/
"ISPEXEC VGET DSN2SSID" ;
if rc = 8 | DSN2SSID = '' then do ;
   retc = DSN2ISPF_MSG('DSN2SSID-Value missing','E') ;
   DSN2RETC = '8' ;
   signal DSN2ISPF_END ;
end ;
DSN2_LOCATION = DSN2ISPF_DB2LOC(DSN2SSID) ; /* Get Location Name      */
if DSN2_LOCATION = '????' then do ;
   retc = DSN2ISPF_MSG('Invalid DSN2SSID-Value: 'DSN2SSID,'E') ;
   DSN2RETC = '8' ;
   signal DSN2ISPF_END ;
end ;

retc = DSN2ISPF_MSG('DSN2SSID set to: 'DSN2SSID,'I') ;
retc = DSN2ISPF_MSG('DSN2_LOCATION set to: 'DSN2_LOCATION,'I') ;
/************************ Select Statement ****************************/
"ISPEXEC VGET DSN2STMT" ;                  /* Determine SELECT-Stmt    */
if rc = 8 | DSN2STMT = '' then do ;
   retc = DSN2ISPF_MSG('DSN2STMT-Value missing','E') ;
   DSN2RETC = '8' ;
   signal DSN2ISPF_END ;
end ;
col_count = Ø ;                            /* Num of Cols in SEL-Stmt  */
db2_colname. = '' ;                        /* Array of Column Names    */
/************************ ISPF KEYLIST ********************************/
"ISPEXEC VGET DSN2KEYS" ;                  /* Determine ISPF Keylist   */
if rc = 8 | DSN2KEYS = '' then DSN2KEYS = '' ;
else do ;
  DSN2_KEYS = translate(DSN2KEYS,' ',',') ;
  key_count = words(DSN2_KEYS) ;
  do kc = 1 to key_count ;
     key_col = word(DSN2_KEYS,kc) ;
     if length(key_col) > 8 then,
     do;
        retc = DSN2ISPF_MSG('Invalid IspKey-Column Name: 'key_col,'E') ;
```

```
        DSN2RETC = '8' ;
        signal DSN2ISPF_END ;
     end;
     col_count = col_count + 1 ;
     db2_colname.col_count = key_col ;
  end kc ;
end ;
/************************ ISPF VAR-NAMES ******************************/
"ISPEXEC VGET DSN2VARS" ;                 /* Determine ISPF Varlist   */
if rc = 8 | DSN2VARS = '' then do ;
   retc = DSN2ISPF_MSG('DSN2VARS-Value missing','E') ;
   DSN2RETC = '8' ;
   signal DSN2ISPF_END ;
end ;
DSN2_VARS = translate(DSN2VARS,' ',',') ;
var_count = words(DSN2_VARS) ;
do vc = 1 to var_count ;
   var_col = word(DSN2_VARS,vc) ;
   if length(var_col) > 8 then,
   do;
      retc = DSN2ISPF_MSG('Invalid Isp-Column Name: 'var_col,'E') ;
      DSN2RETC = '8' ;
      signal DSN2ISPF_END ;
   end;
   col_count = col_count + 1 ;
   db2_colname.col_count = var_col ;
end vc ;
retc =,
DSN2ISPF_MSG('Number of Columns in the Result-Set: 'col_count,'I') ;
/************************ DSN2LOCL ************************************/
"ISPEXEC VGET DSN2LOCL" ;                 /* Determine DSN2LOCL-Value */
if rc = 8 | DSN2LOCL = '' then DSN2LOCL = '1' ;    /* Set default     */
else if DSN2LOCL <> 'Ø' & DSN2LOCL <> '1' then do ;
   retc = DSN2ISPF_MSG('Invalid DSN2LOCL-Value: 'DSN2LOCL,'E') ;
   DSN2RETC = '8' ;
   signal DSN2ISPF_END ;
end ;
/************************ Get local DB2-Subsystem ID ******************/
if DSN2LOCL = '1' then DSN2ISPF_LOCAL_SSID = DSN2ISPF_LOCALDB2() ;
else DSN2ISPF_LOCAL_SSID = DSN2SSID ;
if DSN2ISPF_LOCAL_SSID = '????' then do ;
   retc = DSN2ISPF_MSG('Unable to determine DSN2ISPF_LOCAL_SSID','E') ;
   DSN2RETC = '8' ;
   signal DSN2ISPF_END ;
end ;
retc = DSN2ISPF_MSG('LOCAL_SSID is: 'DSN2ISPF_LOCAL_SSID,'I') ;
/************************ Build the FETCH-Stmt ************************/
DSN2ISPF_FETCH_STMT = "FETCH C1 INTO";
do cc = 1 to col_count ;
   if cc < col_count then ,
```

```
      DSN2ISPF_FETCH_STMT = ,
      DSN2ISPF_FETCH_STMT" :"db2_colname.cc" :I"cc"," ;
   else ,
      DSN2ISPF_FETCH_STMT = ,
      DSN2ISPF_FETCH_STMT" :"db2_colname.cc" :I"cc ;
end cc ;
/************************ Generate a unique ISPF Tablename ************/
Address TSO "ISPEXEC CONTROL ERRORS RETURN";
DSN2ISPF_GENTAB:
DSN2TNAM = "DSN2"random(1,9999);
DSN2TNAM = overlay(DSN2TNAM,copies('Ø',8));
Address TSO "ISPEXEC TBQUERY "DSN2TNAM;
if rc = 12 then,
   Address TSO "ISPEXEC CONTROL ERRORS CANCEL";
else if rc = Ø then signal DSN2ISPF_GENTAB ; /* Table already exists  */
/************************ Create the ISPF Table ***********************/
if DSN2KEYS = '' then,
   Address TSO "ISPEXEC TBCREATE "DSN2TNAM" NAMES ("DSN2VARS")",
               DSN2ISPP ;
else,
   Address TSO "ISPEXEC TBCREATE "DSN2TNAM" NAMES ("DSN2VARS")",
               "KEYS("DSN2KEYS")",
               DSN2ISPP ;
if rc <> Ø then do ;
   retc = DSN2ISPF_MSG('Unable to create ISPF Table:' rc,'E');
   DSN2RETC = '12' ;
   signal DSN2ISPF_END ;
end ;
/************************ Start DSNREXX Processing  *******************/
'SUBCOM DSNREXX' ;
if rc <> Ø then do;
  s_rc = RXSUBCOM('ADD','DSNREXX','DSNREXX') ;
  if s_rc <> Ø then do ;
     retc = DSN2ISPF_MSG('Unable to add DSNREXX-Environment:' s_rc,'E');
     DSN2RETC = '12' ;
     signal DSN2ISPF_END ;
  end ;
  retc = DSN2ISPF_MSG('DSNREXX-Environment added','I') ;
end;
Address DSNREXX ;
"CONNECT "DSN2ISPF_LOCAL_SSID ;
if rc <> Ø then do ;
   retc = ,
   DSN2ISPF_MSG('Error at LOCAL_SSID Attach: 'rc,'E') ;
   DSN2RETC = '12' ;
   signal DSN2ISPF_END ;
end ;
DSN2ISPF_MSG('Successfull Attach to 'DSN2ISPF_LOCAL_SSID,'I') ;
"EXECSQL CONNECT TO "DSN2_LOCATION ;
if rc <> Ø then do ;
```

```
   retc = ,
   DSN2ISPF_MSG('Error at CONNECT to' DSN2_LOCATION,'E') ;
   retc = DSN2ISPF_SQLERROR() ;
   DSN2RETC = SQLCODE ;
   signal DSN2ISPF_END ;
end;
DSN2ISPF_MSG('Successfull Connect to 'DSN2_LOCATION,'I') ;
DSN2ISPF_PKSET = 'DSNREX'||DSN2ISOL ;
"EXECSQL SET CURRENT PACKAGESET = 'DSNREXUR'" ;
if rc <> Ø then do ;
   retc = ,
   DSN2ISPF_MSG('Error at SET CURRENT PACKAGESET','E') ;
   retc = DSN2ISPF_SQLERROR() ;
   signal DSN2ISPF_END ;
end;
DSN2ISPF_MSG('Packageset 'DSN2ISPF_PKSET' will be used','I') ;
/************************ Declare Cursor ******************************/
"EXECSQL DECLARE C1 CURSOR FOR S1" ;
if rc <> Ø then do ;
   retc = ,
   DSN2ISPF_MSG('Error at DECLARE CURSOR','E') ;
   retc = DSN2ISPF_SQLERROR() ;
   signal DSN2ISPF_END ;
end;
/************************ Prepare Statement ***************************/
"EXECSQL PREPARE S1 FROM :DSN2STMT" ;
if rc <> Ø then do ;
   retc = ,
   DSN2ISPF_MSG('Error at PREPARE STATEMENT','E') ;
   retc = DSN2ISPF_SQLERROR() ;
   signal DSN2ISPF_END ;
end;
/************************ Open Cursor *********************************/
"EXECSQL OPEN C1" ;
if rc <> Ø then do ;
   retc = ,
   DSN2ISPF_MSG('Error at OPEN CURSOR','E') ;
   retc = DSN2ISPF_SQLERROR() ;
   signal DSN2ISPF_END ;
end;
/************************ Fetch Loop **********************************/
num_rows = Ø ;                        /* Number of Rows fetched       */
"EXECSQL "DSN2ISPF_FETCH_STMT ;
do while (SQLCODE = Ø & num_rows < DSN2ROWS ) ;
   num_rows = num_rows + 1 ;
   /* Substitute NULL-Values with Empty-String and strip Outputvars   */
   do cc = 1 to col_count ;
      Rx_Cmd = db2_colname.cc" = strip("db2_colname.cc",,)" ;
      interpret Rx_Cmd ;
      Rx_Cmd = "DSN2_NULLVAL = I"cc ;
```

```
      interpret Rx_Cmd ;
      if DSN2_NULLVAL < Ø then do ;
         Rx_Cmd = db2_colname.cc" = ''" ;
         interpret Rx_Cmd ;
      end ;
   end cc ;
   Address TSO "ISPEXEC TBADD "DSN2TNAM ;
   if rc <> Ø then do ;
      retc = ,
      DSN2ISPF_MSG('Error at TBADD (RowNr.'num_rows')','E') ;
      signal DSN2ISPF_END ;
   end ;
   "EXECSQL "DSN2ISPF_FETCH_STMT ;
end ;
if SQLCODE = Ø then do ;
   retc = ,
   DSN2ISPF_MSG('Max. Number of Rows ('num_rows') reached','I') ;
   DSN2RETC = '1' ;
end ;
else if SQLCODE = 1ØØ then do ;
   retc = ,
   DSN2ISPF_MSG('Number of Rows fetched: 'num_rows,'I') ;
   if num_rows = Ø then do ;
      DSN2RETC = '4' ;
      if DSN2TABB = 'NOTAB' then,                  /* Erase ISPF-Table */
         Address TSO "ISPEXEC TBEND "DSN2TNAM ;
      signal DSN2ISPF_CLOSE_CURSOR ;
   end ;
   else do;
     DSN2RETC = 'Ø' ;
     DSN2ERRM = 'Normal Completion' ;
   end ;
end ;
else do ;
   retc = ,
   DSN2ISPF_MSG('Error at FETCH CURSOR','E') ;
   retc = DSN2ISPF_SQLERROR() ;
   signal DSN2ISPF_END ;
end ;
/************************ Close Cursor and Disconnect *****************/
DSN2ISPF_CLOSE_CURSOR:
"EXECSQL CLOSE C1" ;
"EXECSQL CONNECT RESET" ;
"DISCONNECT" ;
Address TSO "ISPEXEC TBTOP "DSN2TNAM ;
DSN2RSLT = num_rows ;                    /* Set Number of Rows fetched */
/************************ End Processing ******************************/
DSN2ISPF_END:
Address TSO ;
/* Clear all DSN2ISPF-Variables in ISPF-Pool                          */
```

```
"ISPEXEC VERASE (DSN2SSID,DSN2STMT,DSN2VARS,DSN2KEYS,",
"DSN2TRCE,DSN2ELVL,DSN2ROWS,DSN2ISOL,DSN2TABB,DSN2ISPP) BOTH" ;
/* Set Returncode,Errormessage and Number of Rows                    */
"ISPEXEC VPUT (DSN2RETC,DSN2ERRM,DSN2RSLT)" ;
return(DSN2TNAM) ;
/*********************************************************************/
/* DSN2ISPF_MSG: SQL-Errors                                          */
/*********************************************************************/
DSN2ISPF_SQLERROR:
DSN2RETC = SQLCODE ;
DSN2ERRM = SQLERRMC ;
retc = ,
DSN2ISPF_MSG('SQLCODE : 'DSN2RETC,'E') ;
retc = ,
DSN2ISPF_MSG(DSN2ERRM,'E') ;
return(0) ;
/*********************************************************************/
/* DSN2ISPF_MSG: Errors and Informations based on Errorlevel         */
/*********************************************************************/
DSN2ISPF_MSG:
parse arg Msg_String,Msg_Type ;
if DSN2ELVL = '0' then return(0) ;         /* Return immediately     */
if DSN2ELVL = '1' & Msg_Type = 'E' then,   /* Errors only            */
   say 'DSN2ISPF => 'Msg_string ;
else if DSN2ELVL = '2' & ( Msg_Type = 'I' | Msg_Type = 'E') then,
   say 'DSN2ISPF => 'Msg_string ;
if Msg_Type = 'E' then DSN2ERRM = Msg_String ;
return(0) ;
/*********************************************************************/
/* DSN2ISPF_SYNTAX: REXX Errors                                      */
/*********************************************************************/
DSN2ISPF_SYNTAX:
retc = DSN2ISPF_MSG('A REXX Error has occurred ||','E') ;
retc = DSN2ISPF_MSG('Maybe wrong Tracelevel     ','E') ;
DSN2RETC = 8 ;
signal DSN2ISPF_END ;
return(0) ;
/*********************************************************************/
/* DSN2ISPF_DB2LOC: Maintain the following Table with your Location  */
/*                  names !!!                                        */
/*********************************************************************/
DSN2ISPF_DB2LOC:
arg conn_ssid ;
select ;
  when ssid = 'DSN'  then return('your-location-name-for-dsn') ;
  /* Specify all of your DB2-Subsystems and their location-names
     here */
  otherwise return('????') ;
end ;
return(0) ;
```

```
/*********************************************************************/
/* DSN2ISPF_LOCALDB2: Returns the name of a DB2 Subsystem that is    */
/*                    running on the LPAR where this REXX EXEC       */
/*                    executes                                       */
/*********************************************************************/
DSN2ISPF_LOCALDB2:
/* Retrieve SMF-ID of the executing System :                         */
cvt1    = STORAGE(1Ø,4)                    /* pointer to CVT hex     */
cvt2    = BITAND(cvt1,'7FFFFFFF'X)         /* clear high order bit   */
cvt     = C2X(cvt2)                        /* pointer to CVT char    */
cvtsmca = D2X(X2D(cvt)+X2D(C4))            /* CVTSMCA address        */
smca1   = STORAGE(cvtsmca,4)               /* CVTSMCA value=SMCA ptr */
smca2   = BITAND(smca1,'7FFFFFFF'X)        /* clear high order bit   */
smca    = C2X(smca2)                       /* pointer to SMCA char   */
if X2D(smca) = Ø then smfid = '????' ;     /* SMF not on system      */
else do ;
   smcasid = D2X(X2D(smca)+X2D(1Ø)) ;      /* SMCASID address        */
   smfid = STORAGE(smcasid,4) ;            /* SMCASID value = SMFID  */
end ;
/* Apply one db2-subsystem for every LPAR you are running            */
select;
 when( smfid = 'SYS1' ) THEN db2sys='DSN';
 otherwise                  db2sys='????';
end ;
return(db2sys) ;
```

SAMPLE PROGRAM TO DEMONSTRATE THE USAGE OF DSN2ISPF

```
/* Rexx
*/
DSN2STMT = "SELECT DBNAME,NAME FROM SYSIBM.SYSTABLESPACE",
           "WHERE STATUS = 'A' AND DBNAME LIKE 'DBA%'",
           "ORDER BY DBNAME,NAME";

DSN2SSID =  'DSN'   ;
DSN2ROWS =  999999  ;
DSN2VARS =  "DB,TS" ;

/*  ----------------------------------------------------------------*/
/* VPUT Parameters for DSN2ISPF:                                     */
/*  ----------------------------------------------------------------*/
"ISPEXEC VPUT (DSN2SSID,DSN2STMT,DSN2ROWS,DSN2VARS)";

/*  ----------------------------------------------------------------*/
/* Call DSN2ISPF and build ISPF-Table                                */
/*  ----------------------------------------------------------------*/
tstab = DSN2ISPF() ;

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

```
/* VGET Return-Values / Check Return-Values                          */
/*  ----------------------------------------------------------------*/
"ISPEXEC VGET (DSN2RETC,DSN2ERRM,DSN2RSLT)";
if DSN2RETC <> Ø then do ;
    if DSN2RETC < Ø then do ;
       say _pgm time()': DSN2ISPF()-SQLSTMT IS: ' ;
       say _pgm time()': 'DSN2STMT ;
    end ;
    say 'DSN2ISPF()-RETURNCODE IS: 'DSN2RETC ;
    say 'DSN2ISPF()-MESSAGE    IS: 'DSN2ERRM ;
    if DSN2RETC > 4 | DSN2RETC < Ø then exit(12) ;
end ;

tstab_count = DSN2RSLT ;                    /* Number of Rows returned */
/*  ----------------------------------------------------------------*/
/* TBSKIP through the Result-Table                                   */
/*  ----------------------------------------------------------------*/
"ISPEXEC TBSKIP "tstab ;
do while rc = Ø ;
   say "Database: "db" Tablespace: "ts ;
   "ISPEXEC TBSKIP "tstab ;
end ;
exit(Ø) ;
```

Peter Adlersburg
Senior DBA
T-Systems (Austria)

Transfer data utility

Our installation is currently using DB2 UDB for OS/390 and z/OS V7 with production and test subsystems installed. Often we need to copy some groups of related tables from one subsystem to the other.

DB2 REXX applications DRAWR2 and DRAWR3 take a table creator (schema) as input and produce control statements for UNLOAD and LOAD utilities, respectively. Algorithms for calculating the size of work datasets are included in the applications too.

Our internal standard is that tables from the same project have the same creator name (schema), so this is used as the filtering criteria for generating jobs. A prerequisite for running the transfer data

utility is that any group of related tables has the identical structure in both subsystems. Since the unit of transfer is the tablespace (replace option is used), some tables with different schema names that belong to the same tablespace with filtered tables will also be included.

The scenario for transferring data between DB2 subsystems is:

- First you must enter your source DB2 subsystem name and filtering criteria (schema name).
- Option 1 is used to generate JCL for unloading data to sequential datasets.
- After successful execution of the submitted job, enter the target DB2 subsystem name (do not change the filter criteria).
- With Option 2, generate the JCL for a reload (load utility).

Note:

- Option 2 removes the check pending and copy pending status with the repair utility, and if you want to execute check data and/ or image copy utilities, you must remove step0002 from the generating job.
- The required parameters on the main input panel are DB2 subsystem name (DSN or DBT at our installation) and the filter criteria (schema name). The names of DB2 runlib load modules library and plan name for DSNTIAUL program are generated according to the chosen DB2 subsystem.

DRAWR1

```
/* rexx DRAWR1 */
address ispexec 'select panel(DRAWP)'
```

DRAWP

```
)ATTR
%  TYPE(TEXT)
[  TYPE(TEXT) INTENS(LOW)
<  TYPE(INPUT) CAPS(ON)
+  TYPE(TEXT) INTENS(LOW)
```

```
!  TYPE(OUTPUT) INTENS(LOW) CAPS(OFF)
)BODY DEFAULT(]*;)EXPAND($$)
%-$-$- DRAW UNLOAD\RELOAD -$-$-[

+DB2 Subsystem     ===><Z   [
[

+Table Creator     ===><Z       [
[

+Plan    ===>!Z       [

+Runlib  ===>!Z                 [



                1. Generate JCL for UNLOAD utility
                2. Generate JCL for RELOAD utility




+Command ===><Z[

)INIT
  .ZVARS = '(DSN8SSID DSN8CRE PLAN RUNLIB ZCMD)'
  .CURSOR = DSN8SSID
  &PLAN = DSNTIB71
  VGET (DSN8SSID DSN8CRE PLAN RUNLIB) SHARED
  IF (&DSN8SSID = &Z)
    &DSN8SSID = DSN
    &RUNLIB   = DSN71Ø.RUNLIB.LOAD
)PROC
  VER (&ZCMD,RANGE,1,2)
  VER  (&DSN8SSID,NB,LIST,DSN,DBT)
  VER (&DSN8CRE,NONBLANK)
  IF (&DSN8SSID = DBT)
    &RUNLIB   = DBT71Ø.RUNLIB.LOAD
  ELSE
    &RUNLIB   = DSN71Ø.RUNLIB.LOAD
  VPUT (DSN8SSID DSN8CRE PLAN RUNLIB) SHARED
  &ZSEL = TRANS(TRUNC(&ZCMD,'.')
               1,'CMD(DRAWR2)'
               2,'CMD(DRAWR3)'
               '  ',' '
               *,'?')
```

```
)END
```

DRAWR2

```
/* REXX DRAWR2 - UNLOAD UTILITY */
/* TRACE I */
signal on error
Address ISPEXEC
"VGET (DSN8SSID, DSN8CRE PLAN RUNLIB) SHARED"
"CONTROL ERRORS RETURN"
userid   = userid()
tick     = ''''
outdsn   = tick||userid||".DRAWUNL.CNTL"||tick
ADDRESS TSO
If sysdsn(outdsn) = "OK" Then
  "alloc fi(dfile) da("outdsn") shr "
Else Do
  "alloc fi(dfile) da("outdsn") new ",
  "  dsorg(ps) space(1,1) tracks",
  "  recfm(F B) lrecl(132) blksize(27984)"
End
outdsn1  = tick||userid||".SYSIN.PRIV"||tick
If sysdsn(outdsn1) = "OK" Then
  "alloc fi(dfile1) da("outdsn1") shr "
Else Do
  "alloc fi(dfile1) da("outdsn1") new ",
  "  dsorg(ps) space(1,1) tracks",
  "  recfm(F B) lrecl(132) blksize(27984)"
End
queue "//"||userid||"X JOB MSGCLASS=X,CLASS=A,NOTIFY="||,
      userid||",REGION=4M"
queue "//*********************************************"
queue "//* UNLOAD DATA                                "
queue "//*********************************************"
queue "//STEP000001 EXEC PGM=IKJEFT01,DYNAMNBR=20"
queue "//SYSTSPRT DD SYSOUT=*"
queue "//SYSTSIN  DD *"
queue " DSN SYSTEM("||DSN8SSID||")"
queue " RUN  PROGRAM(DSNTIAUL) PLAN("||PLAN||") -"
queue "      LIB('"||RUNLIB||"')"
queue "//SYSPRINT DD SYSOUT=*"
queue "//SYSUDUMP DD SYSOUT=*"
queue "//SYSPUNCH DD DUMMY"
"execio 13 diskw dfile"
signal off error
Address TSO "SUBCOM DSNREXX"
IF RC Then S_RC = RXSUBCOM('ADD','DSNREXX','DSNREXX')
Address DSNREXX 'CONNECT' DSN8SSID
If SQLCODE ¬= 0 Then Call SQLCA
```

```
SYSRECNO = Ø
Dbname = " "
Tsname = " "
SQLSTMT = "SELECT DBNAME, TSNAME, NAME, COLCOUNT + RECLENGTH ",
          "FROM SYSIBM.SYSTABLES ",
          "WHERE CREATOR = '" || DSN8CRE || "' AND ",
                "TYPE = 'T' ",
          "ORDER BY 1, 2, 3"
Address DSNREXX "EXECSQL DECLARE C1 CURSOR FOR S1"
Address DSNREXX "EXECSQL PREPARE S1 FROM :SQLSTMT"
Address DSNREXX "EXECSQL OPEN C1"
Address DSNREXX "EXECSQL FETCH C1 INTO :HVDbname, :HVTsname, :HVName,",
                                       ":HVRecLen"
Do While (SQLCODE = Ø)
  If (HVDbname ¬= Dbname | HVTsname ¬= Tsname) Then Do
    Dbname = HVDbname
    Tsname = HVTsname
    SQLSTMT = "SELECT CREATOR, NAME, COLCOUNT + RECLENGTH ",
              "FROM SYSIBM.SYSTABLES ",
              "WHERE DBNAME = '" || Dbname || "' AND ",
                    "TSNAME = '" || Tsname || "' AND ",
                    "CREATOR <> '" || DSN8CRE || "' AND ",
                    "TYPE = 'T' ",
              "ORDER BY 1, 2"
    Address DSNREXX "EXECSQL DECLARE C2 CURSOR FOR S2"
    Address DSNREXX "EXECSQL PREPARE S2 FROM :SQLSTMT"
    Address DSNREXX "EXECSQL OPEN C2"
    Address DSNREXX "EXECSQL FETCH C2 INTO :HVCreator, :HVName1,",
                                          ":HVRecLen1"
    Do While (SQLCODE = Ø)
      TABLE = STRIP(HVCreator) || "." || STRIP(HVName1)
      queue "//SYSREC"||Right(SYSRECNO,2,'Ø')||,
            " DD UNIT=SYSDA,DISP=(NEW,CATLG),"
      "execio 1 diskw dfile"
      Call process_space Dbname Tsname TABLE HVRecLen1
      queue "//         SPACE=(TRK,("||priqty||","||secqty||"),RLSE),"
      "execio 1 diskw dfile"
      queue "//         DSN="||STRIP(LEFT(Dbname,8))||"."||,
            STRIP(LEFT(Tsname,8))||"."||,
            STRIP(LEFT(HVName1,8))||".PRIV"
      "execio 1 diskw dfile"
      queue "   "||TABLE
      "execio 1 diskw dfile1"
      Address DSNREXX "EXECSQL FETCH C2 INTO :HVCreator, :HVName1,",
                                            ":HVRecLen1"
      SYSRECNO = SYSRECNO + 1
    End
    Address DSNREXX "EXECSQL CLOSE C2"
  End
  TABLE = DSN8CRE || "." || STRIP(HVName)
```

```
  queue "//SYSREC"||Right(SYSRECNO,2,'Ø')||,
        " DD UNIT=SYSDA,DISP=(NEW,CATLG),"
  "execio 1 diskw dfile"
  Call process_space Dbname Tsname TABLE HVRecLen
  queue "//          SPACE=(TRK,("||priqty||","||secqty||"),RLSE),"
  "execio 1 diskw dfile"
  queue "//          DSN="||STRIP(LEFT(Dbname,8))||"."||,
        STRIP(LEFT(Tsname,8))||"."||,
        STRIP(LEFT(HVName,8))||".PRIV"
  "execio 1 diskw dfile"
  queue "   "||TABLE
  "execio 1 diskw dfile1"
  Address DSNREXX "EXECSQL FETCH C1 INTO :HVDbname,:HVTsname,:HVName,",
                                         ":HVRecLen"
  SYSRECNO = SYSRECNO + 1
End
Address DSNREXX "EXECSQL CLOSE C1"
Address DSNREXX "DISCONNECT"
If SQLCODE ¬= Ø Then Call SQLCA
"execio Ø diskw dfile1(finis"
queue "//SYSIN    DD DISP=SHR,DSN="||userid||".SYSIN.PRIV"
"execio 1 diskw dfile"
queue "//"
"execio 1 diskw dfile"
"execio Ø diskw dfile(finis"
"free  fi(dfile)"
"free  fi(dfile1)"
"ispexec edit dataset("outdsn")"
signal on error
"ispexec lmerase dataset("outdsn")"
"ispexec lmerase dataset("outdsn1")"
Exit

SQLCA:
  say "SQLSTATE=" SQLSTATE
  say "SQLWARN =" SQLWARN.Ø || "," ||,
                  SQLWARN.1 || "," ||,
                  SQLWARN.2 || "," ||,
                  SQLWARN.3 || "," ||,
                  SQLWARN.4 || "," ||,
                  SQLWARN.5 || "," ||,
                  SQLWARN.6 || "," ||,
                  SQLWARN.7 || "," ||,
                  SQLWARN.8 || "," ||,
                  SQLWARN.9 || "," ||,
                  SQLWARN.1Ø
  say "SQLERRD =" SQLERRD.1 || "," ||,
                  SQLERRD.2 || "," ||,
                  SQLERRD.3 || "," ||,
                  SQLERRD.4 || "," ||,
```

```
                  SQLERRD.5 || "," ||,
                  SQLERRD.6
  say "SQLERRP =" SQLERRP
  say "SQLERRMC=" SQLERRMC
  say "SQLCODE =" SQLCODE
  Exit 2Ø

error:
  say 'error on line:' sigl '  ,rc:' rc
  Exit

process_space:
  parse arg Dbname Tsname TABLE L
  priqty = 1
  secqty = 1
  SQLSTMT = "SELECT COUNT(*) ",
            "FROM " || TABLE
  Address DSNREXX "EXECSQL DECLARE C3 CURSOR FOR S3"
  Address DSNREXX "EXECSQL PREPARE S3 FROM :SQLSTMT"
  Address DSNREXX "EXECSQL OPEN C3"
  Address DSNREXX "EXECSQL FETCH C3 INTO :HVCount"
  Do While (SQLCODE = Ø)
    If HVCount > Ø Then Do
      priqty = MAX((HVCount * L) % 5ØØØØ, priqty)
      secqty = MAX(priqty % 1Ø, secqty)
    End
    Address DSNREXX "EXECSQL FETCH C3 INTO :HVCount"
  End
  Address DSNREXX "EXECSQL CLOSE C3"
  Return
```

DRAWR3

```
/* REXX DRAWR3 - RELOAD UTILITY */
/* TRACE I */
signal on error
wsqty = 1
wsk = 1
wsm = 1
wse = 1ØØØØØ      /* 1ØØ * number from DISCARDS (1ØØØ) */
J = Ø
Address ISPEXEC
"VGET (DSN8SSID, DSN8CRE) SHARED"
"CONTROL ERRORS RETURN"
userid   = userid()
tick     = ''''
outdsn   = tick||userid||".DRAWRLD.CNTL"||tick
ADDRESS TSO
If sysdsn(outdsn) = "OK" Then
```

```
  "alloc fi(dfile) da("outdsn") shr "
Else Do
  "alloc fi(dfile) da("outdsn") new ",
  "  dsorg(ps) space(1,1) tracks",
  "  recfm(F B) lrecl(132) blksize(27984)"
End
outdsn1  = tick||userid||".DRAWRLD.SYSPUNCH"||tick
If sysdsn(outdsn1) = "OK" Then
  "alloc fi(dfile1) da("outdsn1") shr "
Else Do
  "alloc fi(dfile1) da("outdsn1") new ",
  "  dsorg(ps) space(1,1) tracks",
  "  recfm(F B) lrecl(132) blksize(27984)"
End
SQLTYPE. = "UNKNOWN TYPE"
VCHTYPE  = 448; SQLTYPES.VCHTYPE  = 'VARCHAR'
CHTYPE   = 452; SQLTYPES.CHTYPE   = 'CHAR'
LVCHTYPE = 456; SQLTYPES.LVCHTYPE = 'VARCHAR'
VGRTYP   = 464; SQLTYPES.VGRTYP   = 'VARGRAPHIC'
GRTYP    = 468; SQLTYPES.GRTYP    = 'GRAPHIC'
LVGRTYP  = 472; SQLTYPES.LVGRTYP  = 'VARGRAPHIC'
FLOTYPE  = 480; SQLTYPES.FLOTYPE  = 'FLOAT'
DCTYPE   = 484; SQLTYPES.DCTYPE   = 'DECIMAL'
INTYPE   = 496; SQLTYPES.INTYPE   = 'INTEGER'
SMTYPE   = 500; SQLTYPES.SMTYPE   = 'SMALLINT'
DATYPE   = 384; SQLTYPES.DATYPE   = 'DATE'
TITYPE   = 388; SQLTYPES.TITYPE   = 'TIME'
TSTYPE   = 392; SQLTYPES.TSTYPE   = 'TIMESTAMP'
signal off error
Address TSO "SUBCOM DSNREXX"
IF RC Then S_RC = RXSUBCOM('ADD','DSNREXX','DSNREXX')
Address DSNREXX 'CONNECT' DSN8SSID
If SQLCODE ¬= 0 Then Call SQLCA
SYSRECNO = 0
Dbname = " "
Tsname = " "
SQLSTMT = "SELECT DBNAME, TSNAME, NAME, COLCOUNT + RECLENGTH ",
          "FROM SYSIBM.SYSTABLES ",
          "WHERE CREATOR = '" || DSN8CRE || "' AND ",
                "TYPE = 'T' ",
          "ORDER BY 1, 2, 3"
Address DSNREXX "EXECSQL DECLARE C1 CURSOR FOR S1"
Address DSNREXX "EXECSQL PREPARE S1 FROM :SQLSTMT"
Address DSNREXX "EXECSQL OPEN C1"
Address DSNREXX "EXECSQL FETCH C1 INTO :HVDbname, :HVTsname, :HVName,",
                                       ":HVRecLen"
Do While (SQLCODE = 0)
  If (HVDbname ¬= Dbname | HVTsname ¬= Tsname) Then Do
    Dbname = HVDbname
    Tsname = HVTsname
```

```
  J = J + 1
  LineForRepair.Ø = J
  LineForRepair.J = STRIP(Dbname) || "." || STRIP(Tsname)
  REPRES = "REPLACE"
  SQLSTMT = "SELECT CREATOR, NAME, COLCOUNT + RECLENGTH ",
            "FROM SYSIBM.SYSTABLES ",
            "WHERE DBNAME = '" || Dbname || "' AND ",
                  "TSNAME = '" || Tsname || "' AND ",
                  "CREATOR <> '" || DSN8CRE || "' AND ",
                  "TYPE = 'T' ",
            "ORDER BY 1, 2"
  Address DSNREXX "EXECSQL DECLARE C2 CURSOR FOR S2"
  Address DSNREXX "EXECSQL PREPARE S2 FROM :SQLSTMT"
  Address DSNREXX "EXECSQL OPEN C2"
  Address DSNREXX "EXECSQL FETCH C2 INTO :HVCreator, :HVName1,",
                                        ":HVRecLen1"
  Do While (SQLCODE = Ø)
    TABLE = STRIP(HVCreator) || "." || STRIP(HVName1)
    Call process_space STRIP(HVCreator) STRIP(HVName1) HVRecLen1
    Address DSNREXX "EXECSQL DESCRIBE TABLE :TABLE INTO :SQLDA"
    If SQLCODE ¬= Ø Then Call SQLCA
    K = SYSRECNO + 1
    LineSysRec.Ø = K
    LineSysRec.K = "//SYSREC"||Right(SYSRECNO,2,'Ø')||,
          " DD DISP=SHR,DSN="||STRIP(LEFT(Dbname,8))||"."||,
          STRIP(LEFT(Tsname,8))||"."||,
          STRIP(LEFT(HVName1,8))||".PRIV"
    Call DrawLoad
    Do I = 1 To LINE.Ø
      queue LINE.I
      "execio 1 diskw dfile1"
    End
    REPRES = "RESUME YES"
    Address DSNREXX "EXECSQL FETCH C2 INTO :HVCreator, :HVName1,",
                                          ":HVRecLen1"
    SYSRECNO = SYSRECNO + 1
  End
  Address DSNREXX "EXECSQL CLOSE C2"
End
Else REPRES = "RESUME YES"
TABLE = DSN8CRE || "." || STRIP(HVName)
Call process_space DSN8CRE STRIP(HVName) HVRecLen
Address DSNREXX "EXECSQL DESCRIBE TABLE :TABLE INTO :SQLDA"
If SQLCODE ¬= Ø Then Call SQLCA
K = SYSRECNO + 1
LineSysRec.Ø = K
LineSysRec.K = "//SYSREC"||Right(SYSRECNO,2,'Ø')||,
      " DD DISP=SHR,DSN="||STRIP(LEFT(Dbname,8))||"."||,
      STRIP(LEFT(Tsname,8))||"."||,
      STRIP(LEFT(HVName,8))||".PRIV"
```

```
  Call DrawLoad
  Do I = 1 To LINE.Ø
    queue LINE.I
    "execio 1 diskw dfile1"
  End
  Address DSNREXX "EXECSQL FETCH C1 INTO :HVDbname,:HVTsname,:HVName,",
                                          ":HVRecLen"
  SYSRECNO = SYSRECNO + 1
End
Address DSNREXX "EXECSQL CLOSE C1"
Address DSNREXX "DISCONNECT"
If SQLCODE ¬= Ø Then Call SQLCA
"execio Ø diskw dfile1(finis"
wsqty = MAX((wsqty / 4) % 16384, 1)
wssqty = MAX(wsqty % 1Ø, 1)
wsk = MAX(wsk % 16384, 1)
wse = MAX(wse % 16384, 1)
wsse = MAX(wse % 1Ø, 1)
wsk = MAX(wsk, wse)
wssk = MAX(wsk % 1Ø, 1)
wsm = MAX(wsm % 16384, 1)
wssm = MAX(wsm % 1Ø, 1)
queue "//"||userid||"X JOB MSGCLASS=X,CLASS=A,NOTIFY="||,
      userid||",REGION=4M"
queue  "//*********************************************"
queue "//* RELOAD DATA                                 "
queue  "//*********************************************"
queue "//STEPØØØ1 EXEC DSNUPROC,PARM='"||DSN8SSID||",DRAW3'"
queue "//DSNTRACE DD SYSOUT=*"
queue "//SORTLIB  DD DISP=SHR,DSN=SYS1.SORTLIB"
queue "//SORTOUT  DD DSN="||userid||".SORTOUT.PRIV,"
queue "//         DISP=(NEW,DELETE,CATLG),"
queue "//         SPACE=(16384,("||wsk||","||wssk||"),,,ROUND),"
queue "//         UNIT=339Ø"
queue "//SYSUT1   DD DSN="||userid||".SYSUT1.PRIV,"
queue "//         DISP=(NEW,DELETE,CATLG),"
queue "//         SPACE=(16384,("||wsk||","||wssk||"),,,ROUND),"
queue "//         UNIT=339Ø"
queue "//SORTWKØ1 DD DSN="||userid||".SORTWKØ1.PRIV,"
queue "//         DISP=(NEW,DELETE,CATLG),"
queue "//         SPACE=(16384,("||wsqty||","||wssqty||"),,,ROUND),"
queue "//         UNIT=339Ø"
queue "//SORTWKØ2 DD DSN="||userid||".SORTWKØ2.PRIV,"
queue "//         DISP=(NEW,DELETE,CATLG),"
queue "//         SPACE=(16384,("||wsqty||","||wssqty||"),,,ROUND),"
queue "//         UNIT=339Ø"
queue "//SORTWKØ3 DD DSN="||userid||".SORTWKØ3.PRIV,"
queue "//         DISP=(NEW,DELETE,CATLG),"
queue "//         SPACE=(16384,("||wsqty||","||wssqty||"),,,ROUND),"
queue "//         UNIT=339Ø"
```

```
queue "//SORTWKØ4 DD DSN="||userid||".SORTWKØ4.PRIV,"
queue "//          DISP=(NEW,DELETE,CATLG),"
queue "//          SPACE=(16384,("||wsqty||","||wssqty||"),,,ROUND),"
queue "//          UNIT=339Ø"
queue "//SYSDISC  DD DSN="||userid||".SYSDISC.PRIV,"
queue "//          DISP=(NEW,DELETE,CATLG),"
queue "//          SPACE=(16384,("||wse||","||wsse||"),,,ROUND),"
queue "//          UNIT=339Ø"
queue "//SYSERR   DD DSN="||userid||".SYSERR.PRIV,"
queue "//          DISP=(NEW,DELETE,CATLG),"
queue "//          SPACE=(16384,("||wse||","||wsse||"),,,ROUND),"
queue "//          UNIT=339Ø"
queue "//SYSMAP   DD DSN="||userid||".SYSMAP.PRIV,"
queue "//          DISP=(NEW,DELETE,CATLG),"
queue "//          SPACE=(16384,("||wsm||","||wssm||"),,,ROUND),"
queue "//          UNIT=339Ø"
"execio 43 diskw dfile"
Do I = 1 To LineSysRec.Ø
  queue LineSysRec.I
  "execio 1 diskw dfile"
End
queue "//SYSIN    DD DISP=SHR,DSN="||userid||".DRAWRLD.SYSPUNCH"
"execio 1 diskw dfile"
queue  "//*********************************************"
queue "//* REPAIR DATA                                 "
queue  "//*********************************************"
queue "//STEPØØØ2 EXEC DSNUPROC,PARM='"||DSN8SSID||",DRAWR'"||,
      ",COND=(4,LT)"
queue "//DSNUPROC.SYSIN    DD  *"
queue "REPAIR LOG NO"
"execio 6 diskw dfile"
Do I = 1 To LineForRepair.Ø
  queue "  SET TABLESPACE "||LineForRepair.I||" NOCHECKPEND"
  queue "  SET TABLESPACE "||LineForRepair.I||" NOCOPYPEND"
  "execio 2 diskw dfile"
End
queue "//"
"execio 1 diskw dfile"
"execio Ø diskw dfile(finis"
"free  fi(dfile)"
"free  fi(dfile1)"
"ispexec edit dataset("outdsn")"
signal on error
"ispexec lmerase dataset("outdsn")"
"ispexec lmerase dataset("outdsn1")"
Exit

DrawLoad:
  Line.Ø = 3
  Line.1 = "LOAD DATA INDDN SYSREC" || Right(SYSRECNO,2,'Ø')
```

```
Line.2 = "  LOG NO DISCARDS 1000 " || REPRES
Line.3 = "  INTO TABLE " TABLE
Position = 1
Do I = 1 To SQLDA.SQLD
  If I = 1 Then
    Line = " ("
  Else
    Line = " ,"
  Line = Line Left('"'SQLDA.I.SQLNAME'"',20)
  Type  = SQLDA.I.SQLTYPE
  Null  = Type // 2
  If Null Then Type = Type - 1
  Len   = SQLDA.I.SQLLEN
  Prcsn = SQLDA.I.SQLLEN.SQLPRECISION
  Scale = SQLDA.I.SQLLEN.SQLSCALE
  If (Type = DCTYPE   ) Then Do
    Len  = (PRCSN + 2) % 2
    Line = Line "POSITION("RIGHT(POSITION,5)":",
                 RIGHT(POSITION+Len-1,5)")"
  End
  Else
    Line = Line "POSITION("RIGHT(POSITION,5)")"
  Select
  When (Type = CHTYPE | Type = GRTYP) Then
    Type = SQLTYPES.Type"("STRIP(LEN)")"
  When (Type = FLOTYPE  ) Then
    Type = SQLTYPES.Type"("STRIP((LEN*4)-11) ")"
  When (Type = DATYPE | Type = TITYPE | Type = TSTYPE) Then
    Type = SQLTYPES.Type "EXTERNAL("STRIP(LEN)")"
  Otherwise
    Type = SQLTYPES.Type
  End
  If (Type = GRTYP | Type = VGRTYP | Type = LVGRTYP) Then
    Len = Len * 2
  If (Type = VCHTYPE | Type = LVCHTYPE | Type = VGRTYP |,
      Type = LVGRTYP) Then
    Len = Len + 2
  Line = Line Type
  L = Line.0 + 1
  Line.0 = L
  Line.L = Line
  If Null = 1 Then Do
    Line = "  "
    Line = Line Left('',20)
    Line = Line "  NULLIF("RIGHT(POSITION+Len,5)")='?'"
    Position = Position + 1
    L = Line.0 + 1
    Line.0 = L
    Line.L = Line
  End
```

```
    Position = Position + Len
  End I
  L = Line.Ø + 1
  Line.Ø = L
  Line.L = " )"
  Return

SQLCA:
  say "SQLSTATE=" SQLSTATE
  say "SQLWARN =" SQLWARN.Ø || "," || SQLWARN.1 || "," ||,
                  SQLWARN.2 || "," || SQLWARN.3 || "," ||,
                  SQLWARN.4 || "," || SQLWARN.5 || "," ||,
                  SQLWARN.6 || "," || SQLWARN.7 || "," ||,
                  SQLWARN.8 || "," || SQLWARN.9 || "," ||,
                  SQLWARN.1Ø
  say "SQLERRD =" SQLERRD.1 || "," || SQLERRD.2 || "," ||,
                  SQLERRD.3 || "," || SQLERRD.4 || "," ||,
                  SQLERRD.5 || "," || SQLERRD.6
  say "SQLERRP =" SQLERRP
  say "SQLERRMC=" SQLERRMC
  say "SQLCODE =" SQLCODE
  Exit 2Ø

error:
say 'error on line:' sigl '  ,rc:' rc
Exit

process_space:
  parse arg Tbcre Tbname Ltab
  wspqty = 1
  SQLSTMT = "SELECT COUNT(*) ",
            "FROM " || TABLE
  Address DSNREXX "EXECSQL DECLARE C3 CURSOR FOR S3"
  Address DSNREXX "EXECSQL PREPARE S3 FROM :SQLSTMT"
  Address DSNREXX "EXECSQL OPEN C3"
  Address DSNREXX "EXECSQL FETCH C3 INTO :HVCount"
  wsHVCount = Ø
  Do While (SQLCODE = Ø)
    wsHVCount = HVCount
    If HVCount > Ø Then Do
      wspqty = MAX(HVCount * Ltab * 2, wspqty)
    End
    Address DSNREXX "EXECSQL FETCH C3 INTO :HVCount"
  End
  Address DSNREXX "EXECSQL CLOSE C3"
  wsqty = MAX(wspqty, wsqty)
  SQLSTMT = "SELECT A.NAME, MAX(C.COLTYPE), SUM(",
    "CASE WHEN C.COLTYPE = 'GRAPHIC' THEN C.LENGTH * 2 ",
         "WHEN C.COLTYPE = 'VARGRAPHIC' THEN C.LENGTH * 2 + 2 ",
         "WHEN C.COLTYPE = 'VARCHAR' THEN C.LENGTH + 2 ",
```

```
        "WHEN C.COLTYPE = 'DECIMAL' THEN (C.LENGTH + 2) / 2 ",
        "ELSE C.LENGTH ",
  "END), 'I' ",
"FROM SYSIBM.SYSINDEXES A, SYSIBM.SYSKEYS B, SYSIBM.SYSCOLUMNS C ",
"WHERE A.TBCREATOR = '" || Tbcre || "' AND ",
      "A.TBNAME = '" || Tbname || "' AND ",
      "A.CREATOR = B.IXCREATOR AND ",
      "A.NAME    = B.IXNAME AND ",
      "A.TBCREATOR = C.TBCREATOR AND ",
      "A.TBNAME   = C.TBNAME AND ",
      "B.COLNAME = C.NAME ",
"GROUP BY A.NAME ",
"UNION ",
"SELECT A.RELNAME, MAX(C.COLTYPE), SUM(",
  "CASE WHEN C.COLTYPE = 'GRAPHIC' THEN C.LENGTH * 2 ",
        "WHEN C.COLTYPE = 'VARGRAPHIC' THEN C.LENGTH * 2 + 2 ",
        "WHEN C.COLTYPE = 'VARCHAR' THEN C.LENGTH + 2 ",
        "WHEN C.COLTYPE = 'DECIMAL' THEN (C.LENGTH + 2) / 2 ",
        "ELSE C.LENGTH ",
  "END), 'F' ",
"FROM SYSIBM.SYSFOREIGNKEYS A, SYSIBM.SYSCOLUMNS C ",
"WHERE A.CREATOR = '" || Tbcre || "' AND ",
      "A.TBNAME = '" || Tbname || "' AND ",
      "A.CREATOR = C.TBCREATOR AND ",
      "A.TBNAME  = C.TBNAME AND ",
      "A.COLNAME = C.NAME ",
"GROUP BY A.RELNAME"
Address DSNREXX "EXECSQL DECLARE C4 CURSOR FOR S4"
Address DSNREXX "EXECSQL PREPARE S4 FROM :SQLSTMT"
Address DSNREXX "EXECSQL OPEN C4"
Address DSNREXX "EXECSQL FETCH C4 INTO :HVFil1,:HVFil2,:HVL,:HVInd"
keyext = 1
lidx = Ø
lfk  = Ø
Do While (SQLCODE = Ø)
  keyext = keyext + 1
  If HVInd = "I" Then lidx = MAX(HVL, lidx)
  Else lfk = MAX(HVL, lfk)
  Address DSNREXX "EXECSQL FETCH C4 INTO :HVFil1,:HVFil2,:HVL,:HVInd"
End
Address DSNREXX "EXECSQL CLOSE C4"
keyext = keyext * wsHVCount
wsk = MAX(MAX(lidx + 14, lfk + 14) * keyext, wsk)
wsm = MAX(21 * wsHVCount, wsm)
Return
```

Nikola Lazovic
DB2 System Administrator
Postal Savings Bank (Serbia and Monte Negro)

Consolidate messages in DB2 master log

This REXX program finds the DB2 messages in the DB2 master address space log. These messages are consolidated for each hour period and printed. Only the first message text line is assumed for the message code and this line is printed for the message description. In the final report, all these DB2 messages are summarized. If you want to use this utility, you must change DBP1 to your DB2 member name and change 'SYSPDBA' high-level qualifier to your HLQ.

DB2MSGA

```
/* rexx                                                              */
/*********************************************************************/
/* main                                                              */
/* This program counts reports  DB2 messages in the DB2 master log.*/
/*********************************************************************/
arg dbmid
  $free  file(db2log)$
 $alloc fi(db2log) da(syspdba.ps0.$dbmid$.log.temp) shr$
 eof = '0'
 i = 0
 j = 0
 k = 0
 tt = 0
 mf = '0'
 mc1 = 0
 fd = '1'
 call w_print_header
 call w_init_array
 $execio 1 diskr db2log$
 do while eof = '0'
     pull line
     wmsg = word(line,3)
     wday = word(line,4)
     whour= word(line,1)
     wdes = substr(line,29,50)
     if wday = 'MONDAY,'    then wd = '1'
     if wday = 'TUESDAY,'   then wd = '1'
     if wday = 'WEDNESDAY,' then wd = '1'
     if wday = 'THURSDAY,'  then wd = '1'
     if wday = 'FRIDAY,'    then wd = '1'
     if wday = 'SATURDAY,'  then wd = '1'
```

```
      if wday = 'SUNDAY,'    then wd = '1'
      if wd = '1' & fd ='Ø' then call w_print_dtl
      if wd = '1' then wdate = substr(line,35,12)
      if wd = '1' & fd = '1' then fd = 'Ø'
      if wd = '1' then wd = 'Ø'
      if substr(wmsg,1,3) ='DSN' ,
         & substr(wmsg,8,1) = 'I' then
       do
           wh = Ø
           wh = substr(whour,1,2)
           wh = wh + 1
          mf = 'Ø'
          do i = 1 to h.wh
             if mc.wh.i = wmsg then
                do
                  mn.wh.i = mn.wh.i + 1
                  mf = '1'
                end
             end
        if mf = 'Ø' then
             do
                h.wh = h.wh + 1
                k = h.wh
                mc.wh.k = wmsg
                mn.wh.k = 1
                md.wh.k = wdes
             end
       end
      $execio 1 diskr db2log$
      if word(line,2) = '//STARTING' then
              eof = '1'
      if rc > Ø then eof = '1'
end
  call w_print_dtl
  call w_print_total
  $free  file(db2log)$
  exit
w_print_header :
  say '      MEMBER FOR ' dbmid 'MESSAGE OUTPUT '
  say ' D A T E    ' 'HOUR ' 'MSG.CODE' 'COUNT' 'MESSAGE DESC.'
  say '============' '=====' '========' '=====' '======================'
  return
w_print_dtl :
  wp1 ='1'
  do i = 1 to 24
  wp = '1'
  if h.i > Ø then
    do k = 1 to h.i
```

```
    if i < 1Ø then
          w1h = 'Ø' || i-1 || '-' || 'Ø' || i
       else
       if i = 1Ø then
          w1h = 'Ø' || i-1 || '-' || i
        else
          w1h =  i-1 || '-' || i
       if mn.i.k < 1Ø then w1mn = '   ' || mn.i.k
           else if mn.i.k < 1ØØ then w1mn = '  ' || mn.i.k
              else if mn.i.k < 1ØØØ then w1mn = ' ' || mn.i.k
                  else if mn.i.k < 1ØØØØ then w1mn = ' ' || mn.i.k
       if wp1 = '1' then
         say wdate w1h mc.i.k   w1mn   md.i.k
       else if wp = '1' & wp1 = 'Ø' then
            say '            ' w1h mc.i.k  w1mn   md.i.k
          else
               say '                  '  mc.i.k,
                    w1mn   md.i.k
       wp = 'Ø'
       wp1 = 'Ø'
     end
  end
  call fill_result_array1
  call w_init_array
  return
w_print_total:
  say '  '
  say '  '
  say '             T o t a l  C o u n t s     '
  say '  ' 'MSG.CODE' 'COUNT' 'MESSAGE DESC.'
  say '  ' '========' '=====' '======================'
  do i = 1 to tt
       if t1_mn.i < 1Ø then w1mn = '   ' || t1_mn.i
           else if t1_mn.i < 1ØØ then w1mn = '  ' || t1_mn.i
              else if t1_mn.i < 1ØØØ then w1mn = ' ' || t1_mn.i
                  else if t1_mn.i < 1ØØØØ then w1mn = ' ' || t1_mn.i
            say '  ' t1_mc.i  w1mn   t1_md.i
  end
  return
w_init_array:
  do i = 1 to 24
       h.i = Ø
  end
  return
fill_result_array1:
  do i = 1 to 24
    do k = 1 to h.i
    w_tt = 'Ø'
```

```
    if tt > Ø then
       do
           do j = 1 to tt
              if t1_mc.j = mc.i.k then
                  do
                    t1_mn.j = t1_mn.j + mn.i.k
                    w_tt = '1'
                  end
           end
          if w_tt = '0' then
            do
                tt = tt + 1
                t1_mc.tt = mc.i.k
                t1_mn.tt = mn.i.k
                t1_md.tt = md.i.k
            end
       end
       else
        do
          tt = tt + 1
          t1_mc.tt = mc.i.k
          t1_mn.tt = mn.i.k
          t1_md.tt = md.i.k
        end
  end
 end
 return
```

JCL

```
//DB2MSGA   JOB (ACCT),'DB2-DB2LOGA',
//     CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//*
//*****************************************************************
//* DELETE SYSPDBA.PSØ.DBP1.LOG.TEMP DATASET                      **
//*****************************************************************
//DELPDS EXEC PGM=IEFBR14
//DELLOG1 DD   DSN=SYSPDBA.PSØ.DBP1.LOG.TEMP,DISP=(MOD,DELETE,DELETE),
//         SPACE=(CYL,(1Ø,1Ø))
//*
//*****************************************************************
//* DEFINE SYSPDBA.PSØ.DBP1.LOG.TEMP      DATASET                 **
//*****************************************************************
//DEFPDS EXEC PGM=IEFBR14
//DEFLOG1  DD DISP=(NEW,CATLG,DELETE),
//          DSN=SYSPDBA.PSØ.DBP1.LOG.TEMP,
//          SPACE=(CYL,(5Ø,5Ø)),DCB=(RECFM=FB,LRECL=132)
```

```
/*
//*****************************************************************
//* LOAD DBP1MSTR LOG RECORD FROM SDSF TO DATASET                 **
//*****************************************************************
//SDSGET  EXEC PGM=ISFAFD
//ISFOUT  DD SYSOUT=*
//ISFIN DD *
SYSNAME PX*
PREFIX DBP1MSTR
OWNER *
DA
FIND 'DBP1MSTR'
++S
PRINT ODSN 'SYSPDBA.PSØ.DBP1.LOG.TEMP' * SHR
PRINT 1 9999999
PRINT CLOSE
/*
//*****************************************************************
//* RUN LOG ANALYSE REXX PROGRAM FOR DBP1                         **
//*****************************************************************
//DB2DBP1   EXEC PGM=IKJEFTØ1,DYNAMNBR=3Ø,REGION=4Ø96K
//STEPLIB   DD   DSN=ISP.SISPLOAD,DISP=SHR
//SYSEXEC   DD   DSN=SYSPDBA.REXXLIB,DISP=SHR
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN   DD   *
 PROFILE NOPREFIX
 %DB2MSGA DBP1
/*
```

OUTPUT

```
1READY
  PROFILE NOPREFIX
 READY
  %DB2MSGA DBP1
 IKJ56247I FILE DB2LOG NOT FREED, IS NOT ALLOCATED


MEMBER FOR  DBP1 MESSAGE OUTPUT

D A T E     HOUR  MSG.CODE COUNT MESSAGE DESC.
============ ===== ======== ===== ======================
3Ø MAR 2ØØ3 Ø3-Ø4 DSNZØØ2I    1  -DBP1 DSNZINIT SUBSYSTEM DBP1 SYSTEM
                                                  PARAMETERS L
                  DSN75Ø7I    1  -DBP1 DSN7LSTK
                  DSNYØØ1I    1  -DBP1 SUBSYSTEM STARTING
                  DSNJ127I    1  -DBP1 SYSTEM TIMESTAMP FOR BSDS= Ø3.Ø89
```

```
                                                    Ø2:25:27.
        DSNJØØ1I    2  -DBP1 DSNJWØØ7 CURRENT COPY 1 ACTIVE
                                                          LOG
        DSNJØ99I    1  -DBP1 LOG RECORDING TO COMMENCE WITH
        DSNRØØ1I    1  -DBP1 RESTART INITIATED
        DSNRØØ3I    1  -DBP1 RESTART...PRIOR CHECKPOINT
                                                 RBA=Ø76FD72A7AB4
        DSNRØØ4I    1  -DBP1 RESTART...UR STATUS COUNTS
        DSNRØØ5I    1  -DBP1 RESTART...COUNTS AFTER FORWARD
        DSNRØØ6I    1  -DBP1 RESTART...COUNTS AFTER BACKWARD
        DSNRØØ2I    1  -DBP1 RESTART COMPLETED
        DSNLØØ3I    1  -DBP1 DDF IS STARTING
        DSNTØ4I     1  -DBP1 SYSIBM.DSNRLSTØ1 HAS BEEN STARTED
                                                   FOR THE R
        DSN9Ø22I    3  -DBP1 DSNTCSTR 'START RLIMIT' NORMAL
                                                  COMPLETION
        DSNV434I    1  -DBP1 DSNVRP NO POSTPONED ABORT THREADS
                                                       FOUND
        DSNL519I    2  -DBP1 DSNLILNR TCP/IP SERVICES
                                                   AVAILABLE
        DSNLØØ4I    1  -DBP1 DDF START COMPLETE
        DSNL51ØI  189  -DBP1 DSNLVPCS CONVLIMIT NEGOTIATED
        DSNIØ31I    2  -DBP1 DSNILKES - LOCK ESCALATION HAS
        DSN32Ø1I    2  -DBP1 ABNORMAL EOT IN PROGRESS FOR
Ø4-Ø5   DSNJØØ2I    2  -DBP1 FULL ACTIVE LOG DATA SET
        DSNJØØ1I    2  -DBP1 DSNJW3Ø7 CURRENT COPY 1 ACTIVE
                                                          LOG
        DSNJ311I    1  -DBP1 DSNJCØØ5 ASYNCHRONOUS LOG ARCHIVE
        DSNJ354I    1  -DBP1 DSNJCØØ5 ARCHIVE LOG:  ALL ACTIVE
        DSNJ359I    1  -DBP1 MEMBER RESPONSES:

        DSNJØØ3I    2  -DBP1 DSNJOFF3 FULL ARCHIVE LOG VOLUME
        DSNJ139I    1  -DBP1 LOG OFFLOAD TASK ENDED
        DSN32Ø1I    2  -DBP1 ABNORMAL EOT IN PROGRESS FOR
        DSNB3Ø2I    1  -DBP1 DSNB1GC1 GROUP BUFFER POOL GBP4
                                                            IS
        DSNB315I    1  -DBP1 DSNB1GC1 GROUP BUFFER POOL GBP4
                                                            IS
Ø5-Ø6   DSNW133I    3  -DBP1 DSNWVOPX - TRACE DATA LOST, OP1
                                                   NOT ACCESS
        DSNW123I    3  -DBP1 DSNWVOPX - TRACE RECORDING HAS
                                                 BEEN RESUMED
        DSNIØ31I    1  -DBP1 DSNILKES - LOCK ESCALATION HAS
        DSNT376I   16  -DBP1 PLAN=HVLEPLØ1 WITH
        DSNT5Ø1I   16  -DBP1 DSNILMCL RESOURCE UNAVAILABLE
        DSNJØØ2I   14  -DBP1 FULL ACTIVE LOG DATA SET
        DSNJØØ1I   14  -DBP1 DSNJW3Ø7 CURRENT COPY 1 ACTIVE
                                                        LOG
```

```
                  DSNJØØ3I   14  -DBP1 DSNJOFF3 FULL ARCHIVE LOG VOLUME
                  DSNJ139I    6  -DBP1 LOG OFFLOAD TASK ENDED
               Ø6-Ø7 DSNJØØ2I   12  -DBP1 FULL ACTIVE LOG DATA SET
                  DSNJØØ1I   12  -DBP1 DSNJW3Ø7 CURRENT COPY 1 ACTIVE
                                                                  LOG
                  DSNJØØ3I   12  -DBP1 DSNJOFF3 FULL ARCHIVE LOG VOLUME
                  DSNJ139I    6  -DBP1 LOG OFFLOAD TASK ENDED
               Ø7-Ø8 DSNIØ31I    1  -DBP1 DSNILKES - LOCK ESCALATION HAS
               Ø9-1Ø DSNT376I    4  -DBP1 PLAN=BVIZPLØ1 WITH
                  DSNT5Ø1I    4  -DBP1 DSNILMCL RESOURCE UNAVAILABLE
               1Ø-11 DSNL51ØI    1  -DBP1 DSNLVPCS CONVLIMIT NEGOTIATED
1                 DSNJØØ2I    2  -DBP1 FULL ACTIVE LOG DATA SET
                  DSNJØØ1I    2  -DBP1 DSNJW3Ø7 CURRENT COPY 1 ACTIVE
                                                                    LOG
                  DSNJØØ3I    2  -DBP1 DSNJOFF3 FULL ARCHIVE LOG VOLUME
                  DSNJ139I    1  -DBP1 LOG OFFLOAD TASK ENDED
               11-12 DSNJØØ2I    2  -DBP1 FULL ACTIVE LOG DATA SET
                  DSNJØØ1I    2  -DBP1 DSNJW3Ø7 CURRENT COPY 1 ACTIVE
                                                                  LOG
                  DSNJØØ3I    2  -DBP1 DSNJOFF3 FULL ARCHIVE LOG VOLUME
                  DSNJ139I    1  -DBP1 LOG OFFLOAD TASK ENDED
               12-13 DSNJØØ2I    2  -DBP1 FULL ACTIVE LOG DATA SET
                  DSNJØØ1I    2  -DBP1 DSNJW3Ø7 CURRENT COPY 1 ACTIVE
                                                                  LOG
                  DSNJØØ3I    2  -DBP1 DSNJOFF3 FULL ARCHIVE LOG VOLUME
                  DSNJ139I    1  -DBP1 LOG OFFLOAD TASK ENDED
               14-15 DSNL51ØI    4  -DBP1 DSNLVPCS CONVLIMIT NEGOTIATED
               19-2Ø DSNT376I    1  -DBP1 PLAN=ATMRPØØ3 WITH
                  DSNT5Ø1I    1  -DBP1 DSNILMCL RESOURCE UNAVAILABLE
               21-22 DSNW133I   29  -DBP1 DSNWVOPX - TRACE DATA LOST, OP1
                                                      NOT ACCESS
                  DSNW123I   28  -DBP1 DSNWVOPX - TRACE RECORDING HAS
                                                    BEEN RESUMED
                  DSNPØØ7I    1  -DBP1 DSNPXTNØ - EXTEND FAILED FOR
               22-23 DSNW123I    7  -DBP1 DSNWVOPX - TRACE RECORDING HAS
                                                      BEEN RESUMED
                  DSNW133I    6  -DBP1 DSNWVOPX - TRACE DATA LOST, OP1
                                                      NOT ACCESS
                  DSNJØØ2I    2  -DBP1 FULL ACTIVE LOG DATA SET
                  DSNJØØ1I    2  -DBP1 DSNJW3Ø7 CURRENT COPY 1 ACTIVE
                                                                  LOG
                  DSNJØØ3I    2  -DBP1 DSNJOFF3 FULL ARCHIVE LOG VOLUME
                  DSNJ139I    1  -DBP1 LOG OFFLOAD TASK ENDED
               23-24 DSNIØ31I    1  -DBP1 DSNILKES - LOCK ESCALATION HAS
31 MAR 2ØØ3 ØØ-Ø1 DSNT376I   13  -DBP1 PLAN=POSPLNØ1 WITH
                  DSNT5Ø1I   13  -DBP1 DSNILMCL RESOURCE UNAVAILABLE
                  DSNJØØ2I    2  -DBP1 FULL ACTIVE LOG DATA SET
                  DSNJØØ1I    2  -DBP1 DSNJW3Ø7 CURRENT COPY 1 ACTIVE
```

```
                                                                  LOG
            DSNJ311I    1  -DBP1 DSNJC00 5 ASYNCHRONOUS LOG ARCHIVE
            DSNJ354I    1  -DBP1 DSNJC00 5 ARCHIVE LOG:  ALL ACTIVE
            DSNJ359I    1  -DBP1 MEMBER RESPONSES:
            DSNJ003I    2  -DBP1 DSNJOFF3 FULL ARCHIVE LOG VOLUME
            DSNJ139I    1  -DBP1 LOG OFFLOAD TASK ENDED
          01-02 DSNT365I    1  -DBP1    NO DATABASES FOUND
            DSN9022I    1  -DBP1 DSNTDDIS 'DISPLAY DATABASE'
                                                 NORMAL COMPLETI
            DSNI031I    2  -DBP1 DSNILKES - LOCK ESCALATION HAS
            DSNW133I   18  -DBP1 DSNWVOPX - TRACE DATA LOST, OP1
                                                   NOT ACCESS
            DSNJ002I    4  -DBP1 FULL ACTIVE LOG DATA SET
            DSNJ001I    4  -DBP1 DSNJW307 CURRENT COPY 1 ACTIVE
                                                                  LOG
            DSNW123I   18  -DBP1 DSNWVOPX - TRACE RECORDING HAS
                                                     BEEN RESUMED
            DSNJ003I    4  -DBP1 DSNJOFF3 FULL ARCHIVE LOG VOLUME
            DSNJ139I    2  -DBP1 LOG OFFLOAD TASK ENDED
          02-03 DSNT365I    1  -DBP1    NO DATABASES FOUND
            DSN9022I    1  -DBP1 DSNTDDIS 'DISPLAY DATABASE'
                                                 NORMAL COMPLETI
            DSNJ002I   10  -DBP1 FULL ACTIVE LOG DATA SET
            DSNJ001I   10  -DBP1 DSNJW307 CURRENT COPY 1 ACTIVE
                                                                  LOG
            DSNW133I    9  -DBP1 DSNWVOPX - TRACE DATA LOST, OP1
                                                   NOT ACCESS
            DSNJ003I   10  -DBP1 DSNJOFF3 FULL ARCHIVE LOG VOLUME
            DSNJ139I    4  -DBP1 LOG OFFLOAD TASK ENDED
            DSNW123I    9  -DBP1 DSNWVOPX - TRACE RECORDING HAS
                                                     BEEN RESUMED


                T o t a l  C o u n t s
      MSG.CODE COUNT MESSAGE DESC.
      ======== ===== ======================
1     DSNZ002I    1  -DBP1 DSNZINIT SUBSYSTEM DBP1 SYSTEM PARAMETERS L
      DSN7507I    1  -DBP1 DSN7LSTK
      DSNY001I    1  -DBP1 SUBSYSTEM STARTING
      DSNJ127I    1  -DBP1 SYSTEM TIMESTAMP FOR BSDS= 03.089 02:25:27.
      DSNJ001I   54  -DBP1 DSNJW007 CURRENT COPY 1 ACTIVE LOG
      DSNJ099I    1  -DBP1 LOG RECORDING TO COMMENCE WITH
      DSNR001I    1  -DBP1 RESTART INITIATED
      DSNR003I    1  -DBP1 RESTART...PRIOR CHECKPOINT RBA=076FD72A7AB4
      DSNR004I    1  -DBP1 RESTART...UR STATUS COUNTS
      DSNR005I    1  -DBP1 RESTART...COUNTS AFTER FORWARD
      DSNR006I    1  -DBP1 RESTART...COUNTS AFTER BACKWARD
      DSNR002I    1  -DBP1 RESTART COMPLETED
```

```
   DSNL003I    1  -DBP1 DDF IS STARTING
   DSNT704I    1  -DBP1 SYSIBM.DSNRLST01 HAS BEEN STARTED FOR THE R
   DSN9022I    5  -DBP1 DSNTCSTR 'START RLIMIT' NORMAL COMPLETION
   DSNV434I    1  -DBP1 DSNVRP NO POSTPONED ABORT THREADS FOUND
   DSNL519I    2  -DBP1 DSNLILNR TCP/IP SERVICES AVAILABLE
   DSNL004I    1  -DBP1 DDF START COMPLETE
   DSNL510I  194  -DBP1 DSNLVPCS CONVLIMIT NEGOTIATED
   DSNI031I    7  -DBP1 DSNILKES - LOCK ESCALATION HAS
   DSN3201I    4  -DBP1 ABNORMAL EOT IN PROGRESS FOR
   DSNJ002I   52  -DBP1 FULL ACTIVE LOG DATA SET
   DSNJ311I    2  -DBP1 DSNJC005 ASYNCHRONOUS LOG ARCHIVE
   DSNJ354I    2  -DBP1 DSNJC005 ARCHIVE LOG:  ALL ACTIVE
   DSNJ359I    2  -DBP1 MEMBER RESPONSES:

   DSNJ003I   52  -DBP1 DSNJOFF3 FULL ARCHIVE LOG VOLUME
   DSNJ139I   24  -DBP1 LOG OFFLOAD TASK ENDED
   DSNB302I    1  -DBP1 DSNB1GC1 GROUP BUFFER POOL GBP4 IS
   DSNB315I    1  -DBP1 DSNB1GC1 GROUP BUFFER POOL GBP4 IS
   DSNW133I   65  -DBP1 DSNWVOPX - TRACE DATA LOST, OP1  NOT ACCESS
   DSNW123I   65  -DBP1 DSNWVOPX - TRACE RECORDING HAS BEEN RESUMED
   DSNT376I   34  -DBP1 PLAN=HVLEPL01 WITH
   DSNT501I   34  -DBP1 DSNILMCL RESOURCE UNAVAILABLE
   DSNP007I    1  -DBP1 DSNPXTN0 - EXTEND FAILED FOR
   DSNT365I    2  -DBP1    NO DATABASES FOUND
  READY
 END
```

Ali Ozturk
Database Administrator
Pamukbank (Turkey)

WANT TO SUBSCRIBE?

Each monthlyly issue of *DB2 Update* is packed with ideas for improving *your* DB2 installation – and all of the technical details necessary for *you* to put those ideas into practice.

With *DB2 Update* ***you*** get:

- A practical toolkit of ready-made enhancements, developed and tested in a working environment by DB2 experts throughout the world.
- Tutorial articles on system internals.
- Performance tuning tips and measurements.
- Early user reports on new products and releases.

Plus:

- *DB2 Update* will repay the cost of subscribing many times over!

All for a fraction of the cost of a single training course! So what are *you* waiting for?

- Go to http://www.xephonUSA.com/subscribe now to subscribe!
- Subscribe now and receive 25% off of a 12-month subscription*!

***** using promotional code **D82B31**.

* * *

WANT TO CONTRIBUTE?

DB2 Update is written by technical professionals just like you with a desire to share their expert knowledge with the world.

Xephon is always seeking talented individuals to contribute articles to *DB2 Update* – **and get paid for it!**

If you have insight into how to make DB2 more functional, secure, reliable, user friendly, or to generally improve DB2 performance, please visit http://www.xephonUSA.com/contribute for more details on how you can contribute to this definitive industry publication.