131

DB2

September 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.

No more covering letters

If you haven't ordered a CBPDO lately, you may not know that IBM no longer provides a covering letter on paper. Instead, an MVS Custom-Built Product Delivery Offering (CBPDO) Memo to Users Extension (Memo Extension) is stored in the DOCLIB PDS that is the first file on the first tape volume. This means that it now takes more than simply opening the CBPDO box to answer key questions like:

- Exactly what versions of what products, features, and FMIDs are on the tape(s).
- What PUT level of service is included.

WHAT IS CBPDO?

CBPDO is one of several ways to order IBM software for z/OS. ServerPac is another packaged offering; and you can still get stand-alone product tapes.

CBPDO allows you to order multiple software products and all maintenance to a specified PUT level. You receive it all on one logical, standard label volume that may span several 3480 or other cartridges.

The only disadvantage of CBPDO over stand-alone product tapes is that the file numbers on a CBPDO tape are different from those shown in the program directory for the software product. When reading the program directory, you must check the Memo Extension whenever file numbers are mentioned. The volume serial numbers in the program directory are also incorrect for CBPDO install tapes.

PRINTING MEMO EXTENSION

As you might guess from the Memo Extension name, there is also an MVS CBPDO Memo to Users. Unlike the Memo Extension,

it is not customized to your order, but is a normal paper manual (five pages) with order number GI10-0645. In it, you will find JCL to unload RIMLIB, the second file on the tape, which is a PDS of the JCL to process the tape, including unloading and/or printing the Memo Extension. You can also unload RIMLIB and the Memo Extension with the CBPDO TAPE option of the SMP/E RECEIVE menu.

Once customized, the RIMLIB unload JCL might look like this:

```
//RIMLOD EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//TAPE     DD DISP=SHR,DSN=RIMLIB,UNIT=349Ø,LABEL=(2,SL),VOL=SER=IC8Ø66
//TARGET   DD DSN=&SYSUID..PDO.RIMLIB,DISP=(NEW,CATLG,DELETE),
//            SPACE=(6160,(9Ø,15,15)),
//            DCB=(RECFM=FB,BLKSIZE=6160,LRECL=8Ø)
//SYSIN    DD *
 COPY      INDD=TAPE,OUTDD=TARGET
//
```

In RIMLIB, one of the members of the sample JCL is DOCLOD. It loads the PDS containing Memo Extensions to DASD. Customized, it looks very similar to the RIMLIB unload above:

```
//DOCLOD EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//TAPE     DD DISP=SHR,DSN=DOCLIB,UNIT=349Ø,LABEL=(1,SL),VOL=SER=IC8Ø66
//TARGET   DD DSN=&SYSUID..PDO.DOCLIB,DISP=(NEW,CATLG,DELETE),
//            SPACE=(6160,(12,5,27),,,ROUND),
//            DCB=(RECFM=FBA,BLKSIZE=6160,LRECL=8Ø)
//SYSIN    DD *
 COPY      INDD=TAPE,OUTDD=TARGET
//
```

Once loaded to DASD, you can print it. The DOCLIB PDS is defined as RECFM=FBA, indicating that column 1 is an ASA printer carriage control. Even so, when outputting to a SYSOUT= DD statement, IDCAMS REPRO insists on adding a blank carriage control of its own and shifting each line to the right by one column. IEBGENER, even when replaced by DFSORT or SyncSort, does it properly as in the following example:

```
//IEBGENER EXEC PGM=IEBGENER
//SYSPRINT   DD SYSOUT=*
//SYSIN      DD DUMMY
```

```
//SYSUT1     DD DSN=&SYSUID..PDO.DOCLIB(MEMOEXT),DISP=SHR
//SYSUT2     DD SYSOUT=N,DEST=C1Ø4LM
//
```

RIMLIB member MEMOPRT offers an alternative – direct printing from the tape. You can do things more easily – just open the box and print the Memo Extension in one step – by combining the two jobs above into a single job, maybe even adding the RIMLIB unload as a third job step.

Although I have not tested the print facility to be sure it prints the Memo Extension properly (using the printer carriage controls), it should be pointed out that you can do all this with the on-line (ISPF) interface to SMP/E. Option 5.5.1 prints a lot more: Receive/CBPDO Tape/Load RIMLIB, DOCLIB, and PGMDIR/ PSP files to DASD and print all the members.

PGMDLST

RIMLIB member PGMDLST also may be of interest if all you are interested in is the list of products on the tape, their versions, and/ or the PUT level. Here is what it looks like for DB2 Version 7.1:

```
QDB2C71Ø  DB2 MGT TOOL PKG      7.Ø1.Ø
PDB2C71Ø  DB2 MGT TOOL PKG      7.Ø1.Ø
QDB2R71Ø  DB2 REXX LANG SUP     7.Ø1.Ø
PDB2R71Ø  DB2 REXX LANG SUP     7.Ø1.Ø
QDB2B71Ø  DB2 UDB SERVER        7.Ø1.Ø
PDB2B71Ø  DB2 UDB SERVER        7.Ø1.Ø
QDB2N71Ø  NET.DATA              7.Ø1.Ø
PDB2N71Ø  NET.DATA              7.Ø1.Ø
PPUTØ103
PPUTØ104
PPUTØ105
PPUTØ106
PPUTØ107
PPUTØ108
PPUTØ109
PPUTØ11Ø
PPUTØ111
PPUTØ112
PPUTØ2Ø1
PPUTØ2Ø2
PPUTØ2Ø3
PPUTØ2Ø4
```

```
PPUT0205
PPUT0206
PPUT0207
PPUT0208
PPUT0209
PPUT0210
PPUT0211
PPUT0212
PPUT0301
PPUT0302
PRSU0103
PRSU0104
PRSU0105
PRSU0106
PRSU0107
PRSU0108
PRSU0110
PRSU0111
PRSU0112
PRSU0201
PRSU0202
PRSU0203
PRSU0204
PRSU0205
PRSU0206
PRSU0207
PRSU0208
PRSU0209
PRSU0210
PRSU0211
PRSU0212
PRSU0301
PRSU0302
PPUT9999
```

Given that the Memo Extension is 20 pages, not counting the title page, copyright notices, introduction, and table of contents, it may be faster to find the information in PGMDLST. If you used the JCL in the Memo, you will only have RIMLIB on DASD, and can save yourself the step of unloading the Memo Extension.

Perhaps most intriguing is that PGMDLST is program-friendly, in that it would be easy for a program to read it and determine the software products, versions, and/or PUT level. The PUT level is found by dropping the last record of the member (PPUT9999) and then finding the largest PPUT*nnnn* record. In this case, it is

PPUT0302. The format is yymm, so that translates to February 2003.

SUMMARY

CBPDO tapes can be a big time saver over traditional product install tapes. But it is important to read the Memo Extensions before reading the program directory, as some information in the program directory is incorrect for CBPDO tapes.

All that could change, however, if IBM, as they seem to be hinting, discontinues product install tapes, and only produces CBPDO and other packaged offerings. Hopefully then program directories will be written for CBPDO.

Jon E Pearkins
Adiant Corporation (Canada)

Using DB2 710 Real-Time Statistics (RTS) and DSNACCOR

INTRODUCTION

Since the general availability of DB2 Version 7, IBM has announced a new enhancement to DB2 for OS/390 and z/OS. This new feature is called Real Time Statistics (RTS) and was announced with APARs PQ48447, PQ48448, PQ46859, and PQ56256.

With real-time statistics, DB2 collects statistics on table spaces and index spaces and then periodically writes this information to two user-defined tables:

- SYSIBM.TABLESPACESTATS
- SYSIBM.INDEXSPACESTATS.

These tables are kept in a database named DSNRTSDB, which must be started in order to externalize the statistics that are being held in virtual storage. DB2 will then populate the tables with one row per table space or index space, or one row per partition.

You can then write applications that query the statistics and help you decide when to run REORG, RUNSTATS, or COPY, or enlarge your datasets.

A DB2 stored procedure, DSNACCOR, is supplied by IBM to query the RTS database.

This article describes how to implement RTS and how to use DSNACCOR, the IBM stored procedure, to get 'recommendations' about objects to reorganize.

DSNACCOR IMPLEMENTATION

Implementing Real-Time Statistics (RTS)

The first step is to implement Real-Time Statistics (RTS).

In order to set up RTS, you should:

- Create the real-time statistics objects.
- Set the interval for writing statistics.
- Start the real-time statistics database.

Creating the RTS objects

You should use the SQL statements in member DSNTESS of dataset DSN710.SDSNSAMP as a model for creating the real-time statistics objects:

```
CREATE DATABASE DSNRTSDB CCSID EBCDIC;
CREATE TABLESPACE DSNRTSTS IN DSNRTSDB
   CCSID EBCDIC
   CLOSE NO
   LOCKMAX Ø
   LOCKSIZE ROW
   SEGSIZE 32
   USING STOGROUP SYSDEFLT
```

```
         PRIQTY 16Ø
         SECQTY 16;
CREATE TABLE SYSIBM.TABLESPACESTATS
   (DBNAME             CHAR( 8)  NOT NULL,
    NAME               CHAR( 8)  NOT NULL,
    PARTITION          SMALLINT  NOT NULL,
    DBID               SMALLINT  NOT NULL,
    PSID               SMALLINT  NOT NULL,
    UPDATESTATSTIME    TIMESTAMP NOT NULL WITH DEFAULT,
    TOTALROWS          FLOAT             ,
    NACTIVE            INTEGER           ,
    SPACE              INTEGER           ,
    EXTENTS            SMALLINT          ,
    LOADRLASTTIME      TIMESTAMP         ,
    REORGLASTTIME      TIMESTAMP         ,
    REORGINSERTS       INTEGER           ,
    REORGDELETES       INTEGER           ,
    REORGUPDATES       INTEGER           ,
    REORGUNCLUSTINS    INTEGER           ,
    REORGDISORGLOB     INTEGER           ,
    REORGMASSDELETE    INTEGER           ,
    REORGNEARINDREF    INTEGER           ,
    REORGFARINDREF     INTEGER           ,
    STATSLASTTIME      TIMESTAMP         ,
    STATSINSERTS       INTEGER           ,
    STATSDELETES       INTEGER           ,
    STATSUPDATES       INTEGER           ,
    STATSMASSDELETE    INTEGER           ,
    COPYLASTTIME       TIMESTAMP         ,
    COPYUPDATEDPAGES   INTEGER           ,
    COPYCHANGES        INTEGER           ,
    COPYUPDATELRSN     CHAR(6)   FOR BIT DATA,
    COPYUPDATETIME     TIMESTAMP         )
   IN DSNRTSDB.DSNRTSTS CCSID EBCDIC;

CREATE UNIQUE INDEX SYSIBM.TABLESPACESTATS_IX
   ON SYSIBM.TABLESPACESTATS
      (DBID, PSID, PARTITION)
   CLUSTER CLOSE NO;

CREATE TABLE SYSIBM.INDEXSPACESTATS
   (DBNAME             CHAR( 8)  NOT NULL,
    INDEXSPACE         CHAR( 8)  NOT NULL,
    PARTITION          SMALLINT  NOT NULL,
    DBID               SMALLINT  NOT NULL,
    ISOBID             SMALLINT  NOT NULL,
    PSID               SMALLINT  NOT NULL,
    UPDATESTATSTIME    TIMESTAMP NOT NULL WITH DEFAULT,
    TOTALENTRIES       FLOAT             ,
```

```
      NLEVELS            SMALLINT          ,
      NACTIVE            INTEGER           ,
      SPACE              INTEGER           ,
      EXTENTS            SMALLINT          ,
      LOADRLASTTIME      TIMESTAMP         ,
      REBUILDLASTTIME    TIMESTAMP         ,
      REORGLASTTIME      TIMESTAMP         ,
      REORGINSERTS       INTEGER           ,
      REORGDELETES       INTEGER           ,
      REORGAPPENDINSERT  INTEGER           ,
      REORGPSEUDODELETES INTEGER           ,
      REORGMASSDELETE    INTEGER           ,
      REORGLEAFNEAR      INTEGER           ,
      REORGLEAFFAR       INTEGER           ,
      REORGNUMLEVELS     INTEGER           ,
      STATSLASTTIME      TIMESTAMP         ,
      STATSINSERTS       INTEGER           ,
      STATSDELETES       INTEGER           ,
      STATSMASSDELETE    INTEGER           ,
      COPYLASTTIME       TIMESTAMP         ,
      COPYUPDATEDPAGES   INTEGER           ,
      COPYCHANGES        INTEGER           ,
      COPYUPDATELRSN     CHAR(6)   FOR BIT DATA,
      COPYUPDATETIME     TIMESTAMP         )
    IN DSNRTSDB.DSNRTSTS CCSID EBCDIC;

  CREATE UNIQUE INDEX SYSIBM.INDEXSPACESTATS_IX
    ON SYSIBM.INDEXSPACESTATS
       (DBID, ISOBID, PARTITION)
    CLUSTER CLOSE NO;
```

Setting the interval for writing real-time statistics

You can set the interval for writing real-time statistics using the STATSINT parameter of the DSNZPARM.

The default interval is 30 minutes:

```
DSN6SPRM   RESTART,                                          X
           …                                                 X
           STATSINT=3Ø     write RT Statistics every 3Ø mn   X
           …
```

Starting the RTS database

After you create the real-time statistics database, DB2 puts it into a stopped state.

You need to issue START DATABASE(DSNRTSDB) to explicitly start the database:

```
-DB2S STOP DATABASE(DSNRTSDB)
DSNIØ38I  -DB2S DSNISDB - THE REAL-TIME STATISTICS 979
COLLECTION PROCESS IS ENABLED.
```

What statistics are collected?

Some of the important statistics that are collected for table spaces include total number of rows, number of active pages, and time of last COPY, REORG, or RUNSTATS execution.

Some statistics that may help determine when a REORG is needed include space allocated, extents, number of inserts, updates, or deletes (single or mass) since the last REORG or LOAD REPLACE, number of unclustered inserts, number of disorganized LOBs, and number of overflow records created since last REORG.

There are also statistics to help in determining when RUNSTATS should be executed. These include number of inserts/updates/deletes (single and mass) since the last RUNSTATS execution.

Statistics collected to help with COPY determination include distinct updated pages and changes since the last COPY execution, and the RBA/LRSN of first update since last COPY.

There are also statistics gathered on indexes. Basic index statistics include total number of entries (unique or duplicate), number of levels, number of active pages, space allocated, and extents.

Statistics that help determine when a REORG is needed include time when the last REBUILD, REORG, or LOAD REPLACE occurred. There are also statistics regarding the number of updates, deletes (real or pseudo, single or mass), and inserts (random and those that were after the highest key) since the last REORG or REBUILD.

Process effects on Real-Time Statistics

There are several processes that will have an effect on the Real-

Time Statistics. These include SQL, utilities, and the dropping and creating of objects.

SQL inserts, updates, and deletes will cause the Real-Time Statistics to change. As mentioned previously, there are several counters in the RTS tables that keep track of the number of inserts, updates, and deletes. When these SQL operations are performed, the counters are increased accordingly. If an object is dropped, then the statistics in the RTS tables are removed (provided that the RTS tables are available at the time, otherwise the statistics remain and must be manually removed). Rollbacks will also affect the counters.

Utilities will also have an effect on Real-Time Statistics. A LOAD utility will increase counters in line with the number of records loaded, and changes in the number of active pages, index entries, and extents. With REORG, the statistics updated will be different depending on the type of REORG (SHRLEVEL NONE, REFERENCE, or CHANGE). With SHRLEVEL NONE, the only statistics that get updated are the basic ones – number of rows, active pages, and extents. With SHRLEVEL REFERENCE or CHANGE, more detailed statistics are gathered such as the number of rows inserted (as well as the number inserted in clustering sequence), deleted (singleton or mass), and updated since the last REORG. An index REBUILD will only affect the index statistics. The RUNSTATS utility (run with UPDATE ALL parameter) will update the statistics on the number of insert, updates and deletes since the last RUNSTATS. The COPY utility will update the statistics on the number of updated and changed pages since the last copy was taken, as well as the time of the last copy and LRSN of the first update after the last copy.

Using Real Time Statistics

Now that you have collected and externalized the Real-Time Statistics, you can query the tables on your own.

You could write a query against the TABLESPACESTATS table that will identify when a table space needs to be copied because

more than 30% of the pages have changed since the last image copy was taken:

```
SELECT NAME
FROM SYSIBM.SYSTABLESPACESTATS
WHERE DBNAME = 'DB1' and
((COPYUPDATEDPAGES*100)/NACTIVE)>30
```

You could also query this table to compare the last RUNSTATS timestamp to the timestamp of the last REORG on the same object to determine when RUNSTATS is needed. If the date of the last REORG is more recent than the last RUNSTATS, then it may be time to execute RUNSTATS.

```
SELECT NAME
FROM SYSIBM.SYSTABLESPACESTATS
WHERE DBNAME = 'DB1' and
(JULIAN_DAY(REORGLASTTIME)>JULIAN_DAY(STATSLASTTIME))
```

You can also use a DB2-supplied stored procedure to help with this process, and possibly even work toward automating the whole determination and utility execution process. This stored procedure, DSNACCOR, is a sample procedure which will query the RTS tables and determine which objects need to be reorganized, image copied, or updated with current statistics, those that have taken too many extents, and those which may be in a restricted status.

Implementing the DSNACCOR IBM stored procedure

The DSNACCOR stored procedure is supplied by IBM as a load module.

In order to use it, you should:

- Bind the DSNACCOR package.
- Define to the DB2 subsystem the DSNACCOR stored procedure.

Binding the DSNACCOR package

Statements to bind the DSNACCOR package are supplied in the DSNTIJSG member of DSN710.SDSNSAMP:

```
BIND PACKAGE(DSNACCOR) MEMBER(DSNACCOR) -
      ACTION(REPLACE) ISOLATION(UR) ENCODING(EBCDIC) -
      LIBRARY('DSN71Ø.SDSNDBRM')
```

Creating the DSNACCOR stored procedure

Statements to create DSNACCOR stored procedure are supplied in the DSNTIJSG member of DSN710.SDSNSAMP:

```
CREATE PROCEDURE SYSPROC.DSNACCOR
    ( IN QUERYTYPE          VARCHAR(4Ø)    CCSID EBCDIC
    , IN OBJECTTYPE         VARCHAR(3)     CCSID EBCDIC
    , IN ICTYPE             VARCHAR(1)     CCSID EBCDIC
    , IN STATSSCHEMA        VARCHAR(128)   CCSID EBCDIC
    , IN CATLGSCHEMA        VARCHAR(128)   CCSID EBCDIC
    , IN LOCALSCHEMA        VARCHAR(128)   CCSID EBCDIC
    , IN CHKLVL             INTEGER
    , IN CRITERIA           VARCHAR(4Ø96)  CCSID EBCDIC
    , IN UNUSED             VARCHAR(8Ø)    CCSID EBCDIC
    , IN CRUPDATEDPAGESPCT  INTEGER
    , IN CRCHANGESPCT       INTEGER
    , IN CRDAYSNCLASTCOPY   INTEGER
    , IN ICRUPDATEDPAGESPCT INTEGER
    , IN ICRCHANGESPCT      INTEGER
    , IN CRINDEXSIZE        INTEGER
    , IN RRTINSDELUPDPCT    INTEGER
    , IN RRTUNCLUSTINSPCT   INTEGER
    , IN RRTDISORGLOBPCT    INTEGER
    , IN RRTMASSDELLIMIT    INTEGER
    , IN RRTINDREFLIMIT     INTEGER
    , IN RRIINSERTDELETEPCT INTEGER
    , IN RRIAPPENDINSERTPCT INTEGER
    , IN RRIPSEUDODELETEPCT INTEGER
    , IN RRIMASSDELLIMIT    INTEGER
    , IN RRILEAFLIMIT       INTEGER
    , IN RRINUMLEVELSLIMIT  INTEGER
    , IN SRTINSDELUPDPCT    INTEGER
    , IN SRTINSDELUPDABS    INTEGER
    , IN SRTMASSDELLIMIT    INTEGER
    , IN SRIINSDELUPDPCT    INTEGER
    , IN SRIINSDELUPDABS    INTEGER
    , IN SRIMASSDELLIMIT    INTEGER
    , IN EXTENTLIMIT        INTEGER
    , OUT LASTSTATEMENT     VARCHAR(8Ø12)  CCSID EBCDIC
    , OUT RETURNCODE        INTEGER
    , OUT ERRORMSG          VARCHAR(1331)  CCSID EBCDIC
    , OUT IFCARETCODE       INTEGER
    , OUT IFCARESCODE       INTEGER
    , OUT XSBYTES           INTEGER)
```

```
    PROGRAM TYPE MAIN
    MODIFIES SQL DATA
    EXTERNAL NAME DSNACCOR
    COLLID DSNACCOR
    LANGUAGE C
    RUN OPTIONS 'TRAP(OFF),STACK(,,ANY,)'
    ASUTIME NO LIMIT
    STAY RESIDENT NO
    WLM ENVIRONMENT DB2SWLM                - WLM application environment
    COMMIT ON RETURN NO
    PARAMETER STYLE GENERAL WITH NULLS
    RESULT SETS 2;
```

Creating the Work Load Manager (WLM) application environment

Because DSNACCOR must run in a WLM-established stored procedure address space, you should define to the Work Load Manager an application environment, using its ISPF interface:

```
.------------------------------------------------------------------------.
| Browse                                      Line 00000000 Col 001 072  |
| Command ===>                                          SCROLL ===> PAGE |
| ************************** Top of Data ***************************** |
|                                                                        |
| Appl Environment Name . . DB2SWLM                                      |
| Description . . . . . . . DB2 - WLM AS                                 |
| Subsystem type  . . . . . DB2                                          |
| Procedure name  . . . . . DB2SWLM                                      |
| Start parameters  . . . . APPLENV=DB2SWLM,DB2SSN=DB2S,NUMTCB=8         |
|                                                                        |
|                                                                        |
|                                                                        |
| Limit on starting server address spaces for a subsystem instance:      |
|  No limit                                                              |
|                                                                        |
| ************************* Bottom of Data ************************** |
|                                                                        |
'------------------------------------------------------------------------'
```

You should also add the corresponding DB2SWLM STC to an active PROCLIB:

```
****************************** Top of Data ******************************
//DB2SWLM PROC RGN=0K,APPLENV=DB2SWLM,DB2SSN=DB2S,NUMTCB=8
//IEFPROC EXEC PGM=DSNX9WLM,REGION=&RGN,TIME=NOLIMIT,
//        PARM='&DB2SSN,&NUMTCB,&APPLENV'
//SYSTSPRT DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
```

```
//SYSOUT   DD SYSOUT=*
***************************** Bottom of Data ****************************
```

You should notice that the Resource Recovery Services (RRS) attachment is required for stored procedures that run in a WLM-established address space.

Creating a TEMP database

DSNACCOR creates and uses declared temporary tables.

Therefore, before you can invoke DSNACCOR, you need to create a TEMP database and segmented table spaces in the TEMP database:

```
CREATE DATABASE DB2STEMP  AS TEMP;
CREATE TABLESPACE TEMPØ1  IN DB2STEMP
   CLOSE NO
   LOCKMAX Ø
   SEGSIZE 32
   USING STOGROUP SYSDEFLT
         PRIQTY 16ØØ
         SECQTY 16Ø;
```

DSNACCOR USAGE

The DSNACCCL REXX client program

The stored procedure DSNACCOR can be called by a REXX program.

The output of DSNACCOR provides recommendations by using a predetermined set of criteria in formulas that use the RTS as well as user input for their calculations.

You can specify that the DSNACCOR makes recommendations for everything (COPY, REORG, RUNSTATS, EXTENTS, RESTRICT) or for one or more of your choice, or for specific object types (table spaces or indexes).

More information about DSNACCOR can be found in *Appendix H* of the *DB2 710 Administration Guide*.

```
/* REXX */
PARSE ARG SSID          /* GET THE SSID TO CONNECT TO */
```

```
/*************************************************************/
/* SET UP THE HOST COMMAND ENVIRONMENT FOR SQL CALLS.        */
/*************************************************************/
"SUBCOM DSNREXX"                  /* HOST CMD ENV AVAILABLE? */
IF RC THEN                        /* NO--MAKE ONE            */
   S_RC = RXSUBCOM('ADD','DSNREXX','DSNREXX')
/*************************************************************/
/* CONNECT TO THE DB2 SUBSYSTEM.                             */
/*************************************************************/
ADDRESS DSNREXX "CONNECT" SSID
IF SQLCODE ¬= Ø THEN CALL SQLCA
PROC = 'SYSPROC.DSNACCOR'
P1   = RESTRICT            /* QUERY TYPE */
P1I  = -1                  /* -1 = NULL VALUE */
P2   = ALL                 /* OBJECT TYPE */
P2I  = -1                  /* -1 = NULL VALUE */
P3   = F                   /* IC TYPE */
P3I  = -1                  /* -1 = NULL VALUE */
P4   = SYSIBM
P4I  = -1                  /* -1 = NULL VALUE */
P5   = SYSIBM
P5I  = -1                  /* -1 = NULL VALUE */
P6   = DSNACC
P6I  = -1                  /* -1 = NULL VALUE */
P7   = 8                    /* CHECK LEVEL */
P7I  = Ø                   /* -1 = NULL VALUE */
P8   = LEFT(' ',4Ø96,' ')
P8I  = -1                  /* -1 = NULL VALUE */
P9   = LEFT(' ',ØØ8Ø,' ')
P9I  = -1                  /* -1 = NULL VALUE */
PIØ  = 2Ø
PIØI = -1                  /* -1 = NULL VALUE */
P11  = 1Ø
P11I = -1                  /* -1 = NULL VALUE */
P12  = 7    /* NUMBER OF DAYS SINCE LAST IC */
P12I = -1                  /* -1 = NULL VALUE */
P13  = 1
P13I = -1                  /* -1 = NULL VALUE */
P14  = 1
P14I = -1                  /* -1 = NULL VALUE */
P15  = 5Ø
P15I = -1                  /* -1 = NULL VALUE */
P16  = 2Ø
P16I = -1                  /* -1 = NULL VALUE */
P17  = 1Ø
P17I = -1                  /* -1 = NULL VALUE */
P18  = 1Ø
P18I = -1                  /* -1 = NULL VALUE */
P19  = Ø
```

```
P19I = -1                  /* -1 = NULL VALUE */
P2Ø  = 1Ø
P2ØI = -1                  /* -1 = NULL VALUE */
P21  = 2Ø
P21I = -1                  /* -1 = NULL VALUE */
P22  = 1Ø
P22I = -1                  /* -1 = NULL VALUE */
P23  = 1Ø
P23I = -1                  /* -1 = NULL VALUE */
P24  = Ø
P24I = -1                  /* -1 = NULL VALUE */
P25  = 1Ø
P25I = -1                  /* -1 = NULL VALUE */
P26  = Ø
P26I = -1                  /* -1 = NULL VALUE */
P27  = 2Ø
P27I = -1                  /* -1 = NULL VALUE */
P28  = Ø
P28I = -1                  /* -1 = NULL VALUE */
P29  = Ø
P29I = -1                  /* -1 = NULL VALUE */
P3Ø  = 2Ø
P3ØI = -1                  /* -1 = NULL VALUE */
P31  = Ø
P31I = -1                  /* -1 = NULL VALUE */
P32  = Ø
P32I = -1                  /* -1 = NULL VALUE */
P33  = 1    /* EXTENTS LIMIT */
P33I = -1                  /* -1 = NULL VALUE */

LS = LEFT(' ',8Ø12,' ')
LSI = Ø
SRC =  Ø
SRCI = Ø
EMSG = LEFT(' ',1331,' ')
EMSGI = Ø
IRC = Ø
IRCI = Ø
IRSN = Ø
IRSNI = Ø
X = Ø
XI =Ø

L1 = "(:P1 :P1I, :P2 :P2I, :P3 :P3I, :P4 :P4I, :P5 :P5I,"
L2 = " :P6 :P6I, :P7 :P7I, :P8 :P8I, :P9 :P9I, :P1Ø :P1ØI,"
L3 = " :P11 :P11I, :P12 :P12I, :P13 :P13I, :P14 :P14I, :P15 :P15I,"
L4 = " :P16 :P16I, :P17 :P17I, :P18 :P18I, :P19 :P19I, :P2Ø :P2ØI,"
L5 = " :P21 :P21I, :P22 :P22I, :P23 :P23I, :P24 :P24I, :P25 :P25I,"
L6 = " :P26 :P26I, :P27 :P27I, :P28 :P28I, :P29 :P29I, :P3Ø :P3ØI,"
```

```
L7 = " :P31 :P31I, :P32 :P32I, :P33 :P33,"
L8 = " :LS :LSI, :SRC :SRCI, :EMSG :EMSGI, :IRC :IRCI,"
L9 = " :IRSN :IRSNI, :X :XI)"

L =  L1 || L2 || L3 || L4 || L5 || L6 || L7 || L8 || L9
/*************************************************************/
/* CALL THE STORED PROCEDURE DSNACCOR                        */
/*************************************************************/
ADDRESS DSNREXX "EXECSQL" ,
 "CALL" PROC L
IF SQLCODE < Ø THEN CALL SQLCA
/*************************************************************/
/* GET INFORMATION ABOUT RESULT SETS RETURNED BY THE         */
/* STORED PROCEDURE.                                         */
/*************************************************************/
ADDRESS DSNREXX "EXECSQL DESCRIBE PROCEDURE :PROC INTO :SQLDA"
IF SQLCODE ¬= Ø THEN CALL SQLCA
/*
     DO I = 1 TO SQLDA.SQLD
       SAY "SQLDA."I".SQLNAME ="SQLDA.I.SQLNAME";"
       SAY "SQLDA."I".SQLTYPE ="SQLDA.I.SQLTYPE";"
       SAY "SQLDA."I".SQLLOCATOR ="SQLDA.I.SQLLOCATOR";"
     END I
*/
/*************************************************************/
/* SET UP A CURSOR TO RETRIEVE THE ROWS FROM THE RESULT      */
/* SET.                                                      */
/*************************************************************/
LOC1     = Ø
LOC2     = Ø
ADDRESS DSNREXX ,
 "EXECSQL ASSOCIATE LOCATORS (:LOC1, :LOC2) WITH PROCEDURE :PROC"
IF SQLCODE ¬= Ø THEN CALL SQLCA
/*************************************************************/
/* RETRIEVE AND DISPLAY THE ROWS FROM THE RESULT SET 1       */
/*************************************************************/
SAY "* RESULT SET 1 *"
SEQNO = Ø
TEXT  = " "
ADDRESS DSNREXX "EXECSQL ALLOCATE C1Ø1 CURSOR FOR RESULT SET :LOC1"
IF SQLCODE ¬= Ø THEN CALL SQLCA
CURSOR = 'C1Ø1'
ADDRESS DSNREXX "EXECSQL DESCRIBE CURSOR :CURSOR INTO :SQLDA"
IF SQLCODE ¬= Ø THEN CALL SQLCA
DO I = 1 TO SQLDA.SQLD
  SAY "SQLDA."I".SQLNAME ="SQLDA.I.SQLNAME";"
  SAY "SQLDA."I".SQLTYPE ="SQLDA.I.SQLTYPE";"
  SAY "SQLDA."I".SQLLOCATOR ="SQLDA.I.SQLLOCATOR";"
END I
```

```
DO UNTIL(SQLCODE ¬= Ø)
 ADDRESS DSNREXX "EXECSQL FETCH C1Ø1 INTO :SEQNO, :TEXT"
 IF SQLCODE = Ø THEN
  DO
    SAY "*** FETCH     = OK ***"
    SAY TEXT
  END
END
ADDRESS DSNREXX "EXECSQL CLOSE C1Ø1"
IF SQLCODE ¬= Ø THEN CALL SQLCA
SAY "*** CLOSE     = OK ***"
/*****************************************************************/
/* RETRIEVE AND DISPLAY THE ROWS FROM THE RESULT SET 2           */
/*****************************************************************/
SAY "* RESULT SET 2 *"
ØØ1 = LEFT(' ',ØØØ8,' ')
ØØ2 = LEFT(' ',ØØØ8,' ')
ØØ3 = Ø
ØØ4 = LEFT(' ',ØØØ2,' ')
ØØ5 = LEFT(' ',ØØ36,' ')
ØØ6 = LEFT(' ',ØØØ3,' ')
ØØ7 = LEFT(' ',ØØØ3,' ')
ØØ8 = LEFT(' ',ØØØ3,' ')
ØØ9 = LEFT(' ',ØØØ3,' ')
Ø1Ø = LEFT(' ',ØØ4Ø,' ')
Ø11 = LEFT(' ',ØØ08,' ')
Ø12 = LEFT(' ',ØØ4Ø,' ')
Ø13 = LEFT(' ',ØØ4Ø,' ')
Ø14 = LEFT(' ',ØØ4Ø,' ')
Ø15 = Ø
Ø16 = Ø
Ø17 = Ø
Ø18 = Ø
Ø19 = LEFT(' ',ØØ4Ø,' ')
Ø2Ø = Ø
Ø21 = Ø
Ø22 = Ø
Ø23 = Ø
Ø24 = Ø
Ø25 = Ø
Ø26 = Ø
Ø27 = Ø
Ø28 = Ø
Ø29 = Ø
Ø3Ø = Ø
Ø31 = Ø
Ø32 = Ø
Ø33 = Ø
Ø34 = Ø
```

```
O35 = Ø
O36 = Ø
O37 = Ø
O38 = Ø

LØ = " :OØ1 :OØ1I, :OØ2 :OØ2I, :OØ3 :OØ3I, :OØ4 :OØ4I, :OØ5 :OØ5I,"
L1 = " :OØ6 :OØ6I, :OØ7 :OØ7I, :OØ8 :OØ8I, :OØ9 :OØ9I, :O1Ø :O1ØI,"
L2 = " :O11 :O11I, :O12 :O12I, :O13 :O13I, :O14 :O14I, :O15 :O15I,"
L3 = " :O16 :O16I, :O17 :O17I, :O18 :O18I, :O19 :O19I, :O2Ø :O2ØI,"
L4 = " :O21 :O21I, :O22 :O22I, :O23 :O23I, :O24 :O24I, :O25 :O25I,"
L5 = " :O26 :O26I, :O27 :O27I, :O28 :O28I, :O29 :O29I, :O3Ø :O3ØI,"
L6 = " :O31 :O31I, :O32 :O32I, :O33 :O33I, :O34 :O34I, :O35 :O35I,"
L7 = " :O36 :O36I, :O37 :O37I, :O38 :O38I"

L = LØ || L1 || L2 || L3 || L4 || L5 || L6 || L7

ADDRESS DSNREXX "EXECSQL ALLOCATE C1Ø2 CURSOR FOR RESULT SET :LOC2"
IF SQLCODE ¬= Ø THEN CALL SQLCA
CURSOR = 'C1Ø2'
ADDRESS DSNREXX "EXECSQL DESCRIBE CURSOR :CURSOR INTO :SQLDA"
IF SQLCODE ¬= Ø THEN CALL SQLCA
/*
     DO I = 1 TO SQLDA.SQLD
       SAY "SQLDA."I".SQLNAME ="SQLDA.I.SQLNAME";"
       SAY "SQLDA."I".SQLTYPE ="SQLDA.I.SQLTYPE";"
       SAY "SQLDA."I".SQLLOCATOR ="SQLDA.I.SQLLOCATOR";"
     END I
*/
/* SYSPRINT HEADER */
L1 = "DB2 SUBSYSTEM: " SSID " - " DATE('U') TIME()
L2 = " - DSNACCOR RECOMMENDATIONS"
LINEO.1 = L1 || L2
LINEO.2 = " "
"EXECIO * DISKW SYSPRINT (STEM LINEO."
L1 = "DBNAME   NAME     PART TYPE "
L2 = "STATUS                                IC "
L3 = " RUNSTATS EXTENTS REORG"
LINEO.1 = L1 || L2 || L3
LINEO.2 = LEFT("-",92,"-")
"EXECIO * DISKW SYSPRINT (STEM LINEO."
DO UNTIL(SQLCODE ¬= Ø)
  ADDRESS DSNREXX "EXECSQL FETCH C1Ø2 INTO " L
  IF SQLCODE = Ø THEN
    DO
     DBNAME       = OØ1
     NAME         = OØ2
     PARTITION    = O3
     OBJECTTYPE   = OØ4
     OBJECTSTATUS = OØ5
```

```
     IMAGECOPY    = OØ6
     RUNSTATS     = OØ7
     EXTENTS      = OØ8
     REORG        = OØ9
     L1 = DBNAME NAME PARTITION "   "
     L2 = OBJECTTYPE " " OBJECTSTATUS " "
     L3 = IMAGECOPY RUNSTATS "    " EXTENTS "   " REORG
     LINEO.1 = L1 || L2 || L3
     "EXECIO 1 DISKW SYSPRINT (STEM LINEO."
    END
END
ADDRESS DSNREXX "EXECSQL CLOSE C1Ø2"
IF SQLCODE ¬= Ø THEN CALL SQLCA
ADDRESS DSNREXX "EXECSQL COMMIT"
IF SQLCODE ¬= Ø THEN CALL SQLCA
/*****************************************************************/
/* DISCONNECT FROM THE DB2 SUBSYSTEM.                            */
/*****************************************************************/
ADDRESS DSNREXX "DISCONNECT"
IF SQLCODE ¬= Ø THEN CALL SQLCA
/*****************************************************************/
/* DELETE THE HOST COMMAND ENVIRONMENT FOR SQL. */
/*****************************************************************/
S_RC = RXSUBCOM('DELETE','DSNREXX','DSNREXX') /* REMOVE CMD ENV */
RETURN
/*****************************************************************/
/* ROUTINE TO DISPLAY THE SQLCA */
/*****************************************************************/
SQLCA:
TRACE O
SAY 'SQLCODE ='SQLCODE
SAY 'SQLERRMC ='SQLERRMC
SAY 'SQLERRP ='SQLERRP
SAY 'SQLERRD ='SQLERRD.1',',
    || SQLERRD.2',',
    || SQLERRD.3',',
    || SQLERRD.4',',
    || SQLERRD.5',',
    || SQLERRD.6
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 'SQLSTATE='SQLSTATE
EXIT
```

JCL to call DSNACCCL

```
//STEP1    EXEC PGM=IKJEFTØ1,DYNAMNBR=6Ø
//SYSTSPRT DD SYSOUT=*
//SYSPROC  DD DISP=SHR,DSN=$INSTALL.I99Ø557.DB2.REXX
//SYSPRINT DD SYSOUT=2
//SYSTSIN  DD *
  DSNACCCL DB2S
//*
```

DSNACCCL sample output

```
1DB2 SUBSYSTEM:  DB2S  -  Ø4/Ø7/Ø3 16:14:Ø5 - DSNACCOR RECOMMENDATIONS

 DBNAME   NAME     PART TYPE STATUS           IC  RUNSTATS EXTENTS REORG
 ------------------------------------------------------------------------
 DSNDBØ1  DBDØ1    Ø    TS                    FUL YES      NO      YES
 DSNDBØ1  SYSLGRNX Ø    TS                    FUL YES      NO      YES
 DSNDBØ6  SYSDBASE Ø    TS                    FUL YES      NO      YES
 DSNDBØ6  SYSDBAUT Ø    TS                    FUL YES      NO      YES
 DSNDBØ7  DSN4KØ1  Ø    TS                    FUL YES      NO      YES
 DSNDBØ7  DSN4KØ2  Ø    TS                    FUL YES      NO      YES
 DSNDBØ7  DSN4KØ3  Ø    TS                    FUL YES      NO      YES
 DSNDBØ7  DSN4KØ4  Ø    TS                    FUL YES      NO      YES
 DSAMPLE  TEMP     Ø    TS                    FUL YES      NO      YES
 DSNDBØ1  SYSUTILX Ø    TS                    FUL YES      NO      YES
 DSNDBØ6  SYSCOPY  Ø    TS                    FUL YES      NO      YES
 DSNDBØ6  SYSSTATS Ø    TS                    FUL YES      NO      YES
 DSNDBØ6  SYSHIST  Ø    TS                    FUL YES      NO      YES
 DSNDBØ6  SYSOBJ   Ø    TS                    FUL YES      NO      YES
 DSNDBØ6  SYSPKAGE Ø    TS                    FUL YES      NO      YES
 DSNDBØ1  SPTØ1    Ø    TS                    FUL YES      NO      YES
 DB2STEMP TEMPØ1   Ø    TS                    FUL YES      YES     YES
 DSNDBØ1  SCTØ2    Ø    TS                    FUL YES      NO      YES
 DSNDBØ6  SYSPLAN  Ø    TS                    FUL YES      NO      YES
 PTDB     ALOGFILE Ø    TS                    FUL NO       NO      YES
 PTDB     PTITSPAM Ø    TS                    FUL NO       NO      YES
 PTDB     PTITSPAN Ø    TS                    FUL NO       NO      YES
 PTDB     PTITSPAO Ø    TS                    FUL NO       NO      YES
 PTDB     PTITSPAP Ø    TS                    FUL NO       NO      YES
 PTDB     PTITSPAQ Ø    TS                    FUL NO       NO      YES
 DB2STEMP TEMPØ2   Ø    TS                    FUL YES      NO      YES
 DB2STEMP TEMPØ3   Ø    TS                    FUL YES      NO      YES
 DSNDBØ6  SYSUSER  Ø    TS                    FUL YES      NO      YES
 DSNDBØ1  DSNLLXØ2 Ø    IX                    YES YES      NO      YES
```

DSNDBØ1	DSNLLXØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNDSXØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNDTXØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNDXXØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNDXXØ2	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNATXØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNATXØ2	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNDDHØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNADXØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNADHØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNDCXØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNDTXØ2	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNDKXØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNDXXØ3	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNDRXØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNDPXØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNDDXØ2	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNATXØ3	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNDCXØ2	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNDRXØ2	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNDPXØ2	Ø	IX	YES	YES	NO	YES
DSNDBØ1	DSNLUXØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ1	DSNLUXØ2	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNUCHØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNUCXØ1	Ø	IX	YES	YES	NO	YES
DSAMPLE	IXEMP1	Ø	IX	YES	NO	NO	NO
DSAMPLE	EMPX1	Ø	IX	YES	NO	NO	NO
DSAMPLE	EMPX2	Ø	IX	YES	NO	NO	NO
DSNDBØ6	DSNTNXØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNHHXØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNHDXØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNHHXØ2	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNOFXØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNOFXØ2	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNOFXØ3	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNOFXØ4	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNOFXØ5	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNOFXØ6	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNOFXØ7	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNOAXØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNOAXØ2	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNOAXØ3	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNOPXØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNOPXØ2	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNOPXØ3	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNOFXØ8	Ø	IX	YES	YES	NO	YES
DSNDBØ6	DSNKKXØ1	Ø	IX	YES	YES	NO	YES
DSNDBØ1	DSNSPTØ1	Ø	IX	YES	YES	YES	YES
DSNDBØ1	DSNSPTØ2	Ø	IX	YES	YES	NO	YES

```
DSNDBØ6  DSNKKXØ2 Ø    IX                          YES YES      NO       YES
DSNDBØ6  DSNKSXØ1 Ø    IX                          YES YES      NO       YES
DSNDBØ6  DSNKAXØ1 Ø    IX                          YES YES      NO       YES
DSNDBØ6  DSNKAXØ2 Ø    IX                          YES YES      NO       YES
DSNDBØ6  DSNKAXØ3 Ø    IX                          YES YES      NO       YES
DSNDBØ1  DSNSCTØ2 Ø    IX                          YES YES      NO       YES
DSNDBØ6  DSNAPHØ1 Ø    IX                          YES YES      NO       YES
DSNDBØ6  DSNAPXØ1 Ø    IX                          YES YES      NO       YES
DSNDBØ6  DSNKLXØ1 Ø    IX                          YES YES      NO       YES
DSNDBØ6  DSNKLXØ2 Ø    IX                          YES YES      NO       YES
PTDB     ALOG1MF1 Ø    IX                          YES NO       NO       YES
PTDB     PTPA1V1E Ø    IX                          YES NO       NO       YES
PTDB     PTPA1LI$ Ø    IX                          YES NO       NO       YES
DSNDBØ6  DSNAUHØ1 Ø    IX                          YES YES      NO       YES
DSNDBØ6  DSNAUXØ2 Ø    IX                          YES YES      NO       YES
PTDB     PTITSMG9 Ø    TS   DB=RW ;TS=RW,COPY
```


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 "//STEPØØØ1 EXEC PGM=IKJEFTØ1,DYNAMNBR=2Ø"
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 ¬= Ø 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.0 = 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 1ØØØ " || 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'"',2Ø)
  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.Ø + 1
  Line.Ø = L
  Line.L = Line
  If Null = 1 Then Do
    Line = "  "
    Line = Line Left('',2Ø)
    Line = Line "  NULLIF("RIGHT(POSITION+Len,5)")='?'"
    Position = Position + 1
    L = Line.Ø + 1
    Line.Ø = 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)

The DB2 UDB db2relocatedb command

Have you ever wanted to quickly rename a database in DB2 UDB? If the answer is yes, then did you know about the **db2relocatedb** command? It was available in UDB DB2 7.2, but the tests I did were under DB2 UDB 8.1 FP1 running on Windows 2000 and using the db2admin userid.

It is a relatively simple command, which has as input a parameter file, shown below (taken from the *Release Notes Version7.2/ Version7.1FixPak4* manual):

- DB_NAME=oldName,newName
- DB_PATH=oldPath,newPath
- INSTANCE=oldInst,newInst
- NODENUM=nodeNumber
- LOG_DIR=oldDirPath,newDirPath
- CONT_PATH=oldContPath1,newContPath1
- CONT_PATH=oldContPath2,newContPath2.

This is best shown in an example, so let's try to rename the SAMPLE database in the DB2 instance to a database called HM01 in the same instance. Create a parameter file (called rename01.txt) that contains:

```
DB_NAME=SAMPLE,HMØ1
INSTANCE=DB2
DB_PATH=C:
```

You would then run the command as:

```
>db2relocatedb -f renameØ1.txt
```

If the command execution is successful, you will see:

```
Files and control structures were changed successfully.
Database was catalogued successfully.
DBT1ØØØI  The tool completed successfully.
```

It's as simple as that! Below, I have tried to ask (and answer) some questions that might occur to you:

Can users still be connected to the database when I issue the rename command?

The obvious answer is no!! If you try, you will get something like the following message:

```
DBT1006N  The file/device "C:\DB2\NODE0000\SQL00002\SQLOGCTL.LFH" could
not be opened.
```

Do I need to take a back-up of the database after renaming it if I am using circular logging?

No, you don't have to, but I would so that I could use the Control Center to perform any required recovery.

Do I need to take a back-up of the database after renaming it if I am using archive logging?

No, you don't have to, but I would so that I could use the Control Center to perform any required recovery.

What happens to the database back-up information?

If you have renamed the SAMPLE database to HM01 and then you issue the **>db2 list history backup all for hm01** command, you will see the back-ups you took when the database was called SAMPLE. Note that the location of the back-up will *not* change. So for the HM01 database, the back-up location could be c:\backups\SAMPLE.0\DB2\NODE0000\CATN0000\20030228.

Do *not* delete the back-up directories when you rename a database.

There is no method to update the database name in the back-up path name, so do not try to rename the directory either.

What happens if you try to restore a database copy from a timestamp taken before you did the rename?

This is best discussed in an example. So what we are going to

do is recover from a back-up taken at the old name and roll through the logs from the old and new names. This scenario is shown below:

1 Create the SAMPLE database:

```
>db2sampl
```

2 Turn on archive logging:

```
>db2 update db cfg for sample using logretain on
```

3 Take an offline back-up:

```
>db2 backup db sample to c:\backups
Backup successful. The timestamp for this backup image is:<tmstmp1>
```

4 Update row EMPNO=000010 in the EMPLOYEE table to 53000.00:

```
>db2 connect to sample
>db2 select salary from employee where empno = '000010'
SALARY
-----------
   52750.00
>db2 update employee set salary = salary + 250
>db2 select salary from employee where empno = '000010'
SALARY
-----------
   53000.00
>db2 connect reset
```

5 Rename the SAMPLE database to HM01. The file rename01.txt contains:

```
DB_NAME=SAMPLE,HM01
INSTANCE=DB2
DB_PATH=C:
>db2relocatedb -f rename01.txt
```

6 Update row EMPNO=000010 in the EMPLOYEE table to 53250.00:

```
>db2 connect to hm01
>db2 select salary from employee where empno = '000010'
SALARY
-----------
   53000.00
>db2 update employee set salary = salary + 250
>db2 select salary from employee where empno = '000010'
```

```
SALARY
-----------
   53250.00
```

7 Marker point (take timestamp):

```
>db2 select current time from sysibm.sysdummy1
--------
<timestamp2>
```

8 Update the EMPLOYEE table:

```
>db2 update employee set salary = salary + 250

>db2 select salary from employee where empno = '000010'
SALARY
-----------
   53500.00

>db2 connect reset
```

9 Recover the EMPLOYEE table to point (7):

```
>db2 RESTORE DATABASE SAMPLE FROM c:\backups TAKEN AT <timestamp1>
INTO HM01 WITH 1 BUFFERS BUFFER 1024 REPLACE EXISTING PARALLELISM 1
WITHOUT PROMPTING

>db2 ROLLFORWARD DATABASE HM01 TO <timestamp2> AND COMPLETE

>db2 connect to hm01

>db2 select salary from employee where empno = '000010'
SALARY
-----------
   53250.00
```

The value of SALARY for row EMPNO='000010' after the recovery is the correct value, which shows that you can recover from a renamed database and roll forward through the logs from both databases.

Note that in the above scenario I did not use the Control Center to perform the recovery. If you do, the command that will be generated is:

```
RESTORE DATABASE HM01 FROM "c:\backups" TAKEN AT <timestamp> WITH 1
BUFFERS BUFFER 1024 PARALLELISM 1 WITHOUT PROMPTING;
ROLLFORWARD DATABASE HM01 TO <timestamp2> AND COMPLETE;
```

This will not work, because DB2 will look for a back-up file with HM01 in the pathname! You need to issue the command manually, adding the INTO HM01 and REPLACE EXISTING keywords (as shown in the scenario above).

There is a lot more that you can do with the **db2relocatedb** command. You can:

- Move/copy databases between instances.
- Rename database containers.
- Rename the database log path directory.

Note that all of the above require more than just the issuing of the **db2relocatedb** command! This is shown below.

How do I move a database between instances?

You first need to move the underlying database files to the new instance, then issue the **db2relocatedb** command. As an example, let's say we have a database called SAMPLE in the DB2 instance and we want to copy it to a new instance called PROD. The steps are as follows:

Create the new instance: >db2icrt PROD

From Windows Explorer create a new file called C:\PROD (this isn't created for you when you create the instance).

Create a file under C:\PROD called NODE0000.

Find the database directory name for the SAMPLE database:

```
>db2 list db directory on c:

 Database alias                       = SAMPLE
 Database name                        = SAMPLE
 Database directory                   = SQL00002
 Database release level               = a.00
 Comment                              =
 Directory entry type                 = Home
 Catalog database partition number    = 0
 Database partition number            = 0
```

From Windows Explorer copy everything under file

C:\DB2\NODE0000\SQL00002 to C:\PROD\NODE0000\SQL00002.

From Windows Explorer copy everything under file C:\DB2\NODE0000\SQLDBDIR to C:\PROD\NODE0000\SQLDBDIR.

Create a parameter file (rename02.txt) which contains:

```
DB_NAME=SAMPLE
INSTANCE=DB2,PROD
DB_PATH=C:
```

You have to run the **db2relocatedb** command from the new instance:

```
>set db2instance=prod
>echo %db2instance%
prod
```

Run the **db2relocatedb** command:

```
>db2relocatedb -f rename02.txt
Files and control structures were changed successfully.
Database was catalogued successfully.
DBT1000I  The tool completed successfully.

>db2 list db directory

System Database Directory

 Number of entries in the directory = 1

Database 1 entry:

 Database alias                       = SAMPLE
 Database name                        = SAMPLE
 Database drive                       = C:\PROD
 Database release level               = a.00
 Comment                              = Cataloged by db2relocatedb
 Directory entry type                 = Indirect
 Catalog database partition number    = 0

>db2start

>db2 connect to sample
```

We have shown that we can move a database between instances and connect to it. You should bear in mind that this was the

sample database – a very small one – which uses only default SMS tablespaces! If you are going to move a large database, the copy time may/will be significant.

Note, if you get the message ‘DBT1025N Neither old nor new database name were found in the database directory’, when issuing the **db2relocatedb** command, it means you issued it without first moving the underlying database files!

How do I move a database to a different drive letter?

If you want to move a database from the DB2 instance from the C:\ drive to the D:\ drive, then you need to:

- Create the D:\DB2\NODE0000 directory.
- Create the D:\DB2\NODE0000\SQLDBDIR directory.
- Copy C:\DB2\NODE0000\SQLDBDIR to D:\DB2\NODE0000\SQLDBDIR.
- Copy any SMS directories or DMS files from the C:\ drive to the D:\ drive.
- Create a parameter file (rename04.txt) containing:

```
DB_NAME=HMØ2
INSTANCE=DB2
DB_PATH=C:,D:
```

- Issue the command:

```
>db2relocatedb -f renameØ4.txt
```

You should see:

```
Files and control structures were changed successfully.
Database was catalogued successfully.
DBT1000I The tool completed successfully.
```

If you get the message ‘DBT1006N The file/device "D:\DB2\NODE0000\SQLDBDIR\SQLDBDIR" could not be opened’, it means you didn’t create the D:\DB2 directory before issuing the **db2relocatedb** command.

How do I change a container path name for an SMS table space?

Let's try to change an SMS container path name for a table space in database HM02.

Create the HM02 database:

```
>db2 create db HMØ2
```

For database HM02 create an SMS table space called SMS01:

```
>db2 connect to hmØ2
>db2 CREATE REGULAR TABLESPACE SMSØ1 PAGESIZE 4 K MANAGED BY SYSTEM
USING ('C:\smsØ1' )
```

To check that this has worked, first get the table space ID for the newly created table space:

```
>db2 list tablespaces

 Tablespace ID                        = 3
 Name                                 = SMSØ1
 Type                                 = System managed space
 Contents                             = Any data
 State                                = Øx0000
   Detailed explanation:
     Normal
```

Using the table space ID from above, display the container information:

```
>db2 list tablespace containers for 3

            Tablespace Containers for Tablespace 3

 Container ID                         = Ø
 Name                                 = C:\smsØ1
 Type                                 = Path
```

Create a table in this table space:

```
>db2 create table test (id int) in smsØ1
>db2 insert into test values(4)
>db2 select * from test

ID
-----------
          4

>db2stop force
```

Use Windows Explorer to rename C:\SMS to C:\NEWSMS, which is the name of the new container.

Create a parameter file called rename03.txt containing:

```
DB_NAME=HMØ2
INSTANCE=DB2
DB_PATH=C:
CONT_PATH=C:\SMSØ1,c:\NEWSMS
```

Run the following commands:

```
>db2 connect reset

>db2relocatedb -f renameØ3.txt

Files and control structures were changed successfully.
DBT1000I The tool completed successfully.
>db2start
```

I issued the **db2stop**/**db2start** commands just to make sure that nothing was accessing the database when I was trying to move the file!

If you now issue the list tablespace command shown below, you can see that the name of the container for tablespace id 3 is the new name (NEWSMS):

```
>db2 list tablespace containers for 3

            Tablespace Containers for Tablespace 3

 Container ID                         = Ø
 Name                                 = C:\NEWSMS
 Type                                 = Path

Now make sure you can still access the test table:

>db2start
>db2 connect to hmØ2
>db2 select * from test

ID
-----------
          4
```

We have been able to access our test table in the new container.

If you get a 'DBT1006N The file/device

"c:\NEWSMS\SQLTAG.NAM" could not be opened.' message when running **db2relocatedb**, it means that you did not copy the SQLTAG.NAM file between the directories.

How do I change a container path name for a DMS table space?

You follow the same process as for changing a container path name for an SMS table space, except that instead of renaming the directory you would rename the file C:\DMS01 to C:\NEWDMS (if these were your old/new container names!).

If you get the message 'DBT1006N The file/device "c:\NEWDMS" could not be opened', then you didn't rename the file before issuing the **db2reloactedb** command.

How do I change the log path for a database?

Although the **db2relocatedb** command allows you to change log path names, I would use the:

```
db2 update dg cfg for <db-alias> using newlogpath <new-name>
```

command instead.

I would always make the following general recommendations:

- Make sure that you are the only person accessing the instance.
- Always take an offline back-up when you have finished.

The **db2relocatedb** command is a very good tool for performing tasks that in the past would have been messy (if not difficult). I particularly liked the speed with which I could rename a database. I would always practise before trying this on a live system!

C Leonard
Freelance Consultant (UK)

IBM has announced DB2 Archive Log Accelerator for z/OS Version 2.

DB2 Archive Log Accelerator for z/OS Version 2, previously released as DB2 Archive Log Compression, makes DFSMS hardware compression available to DB2 archive logs, reduces storage and I/O requirements for managing large-volume DB2 archive logs, enables faster writing and reading of archive logs through data striping, allows archive logs to be striped, compressed, or both, simplifies the creation and storage of multiple archive copies, and offers more archive log management options in a single package.

For further information contact your local IBM representative.
URL: http://www.ibm.com/software/data/db2imstools.

* * *

Responsive has announced Cache Xtender (CX), which is designed to optimize the use of DASD Control Unit cache memory for datasets accessed by subsystems such as DB2, IMS/DB, IMS/DBCT, and CICS VSAM.

CX allows system administrators to specify entire performance critical objects, such as DB2 or IMS indices or a partition of an object, to be kept in DASD cache memory. CX is said to be particularly useful when dealing with very large critical indices or data that cannot fit into the host memory pools, but can fit into the DASD cache.

When using CX with DB2 and IMS/DB in a sysplex environment with data sharing, the host buffer pools and CF GBPs may be kept at a more reasonable size by keeping large critical objects fully-cached in the DASD memory.

CX is designed to integrate with any automation product available on z/OS.

For further information contact:
Responsive Systems, 281 Highway 79, Morganville, NJ 07751, USA.
Tel; (732) 972 1261.
URL: http://www.responsivesystems.com/announcements/CX1%20Press%20Release.pdf.

* * *

IBM has announced that it is expanding the DB2 solutions supporting Workgroup Edition, Workgroup Server Edition, and Workgroup Server Unlimited Edition by offering enhanced tools for AIX, Linux, and Windows.

The new support is available in DB2 Table Editor for Workgroups V4.3, DB2 High Performance Unload for Workgroups V2.1, DB2 Performance Expert for Workgroups V1.1, and DB2 Web Query Tool for Workgroups V1.3.

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