65

DB2

March 1998

In this issue

update

DB2 Update

Published by

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

North American office

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

Australian office

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

Contributions

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

Editor

Trevor Eddolls

Disclaimer

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

Subscriptions and back-issues

A year's subscription to *DB2 Update*, comprising twelve monthly issues, costs £235.00 in the UK; $350.00 in the USA and Canada; £241.00 in Europe; £247.00 in Australasia and Japan; and £245.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1994 issue, are available separately to subscribers for £20.00 ($30.00) each including postage.

DB2 Update on-line

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

Printed in England.

Changes to the system catalog for DB2 Version 5

The DB2 catalog is contained in a single database, DSNDB06, and, as of Version 5, consists of 54 tables. These tables collectively describe the objects and resources available to DB2. With each new release of DB2, IBM modifies and tunes the DB2 catalog and directory structure. This is done to enable new capabilities, to extend the features of capabilities available in previous releases of DB2, and for various other tuning and/or performance reasons. DB2 V5 causes modifications to be made to older DB2 catalog tables, as well as adding new tables and indexes.

THE NEW TABLES

Eight tables were added to the DB2 catalog for DB2 Version 5. Prior to DB2 V5, six of these tables were stored in the communication database, also known as the CDB. The CDB was used to describe the connections of a local DB2 subsystem to other systems. The CDB tables were housed in a separate database – DSNDDF. As of V5, the tables were renamed and moved into the DB2 Catalog. The CDB tables affected are as follows:

Old CDB table name	*New DB2 catalog table name*
SYSIBM.SYSLOCATIONS	SYSIBM.LOCATIONS
SYSIBM.SYSLULIST	SYSIBM.LULIST
SYSIBM.SYSLUMODES	SYSIBM.LUMODES
SYSIBM.SYSLUNAMES	SYSIBM.LUNAMES
SYSIBM.SYSMODESELECT	SYSIBM.MODESELECT
SYSIBM.SYSUSERNAMES	SYSIBM.USERNAMES

The definitions for the CDB tables have changed, in some cases substantially.

- SYSIBM.LOCATIONS contains a single row for each accessible server, equating a location with its SNA or TCP/IP network

attributes.

- SYSIBM.LULIST enables you to specify multiple LUNAMES for any given LOCATION. It is used to assign a VTAM LUNAME to a LINKNAME (from SYSIBM.LOCATIONS).
- SYSIBM.LUMODES contains conversation limits for a specific LUNAME/MODENAME combination. It is used to control change-number-of-sessions (CNOS) negotiations at DDF start-up.
- SYSIBM.LUNAMES contains a single row for each LU associated with one or more other systems accessible to the local DB2 subsystem.
- SYSIBM.MODESELECT assigns mode names to conversations supporting outgoing SQL requests.
- SYSIBM.USERNAMES is used to enable outbound and inbound ID translation.

The two other tables added to the DB2 Catalog for DB2 V5 are SYSIBM.IPNAMES and SYSIBM.SYSDUMMY1. SYSIBM.IPNAMES contains a single row for each LU associated with one or

LINKNAME	Must match the LINKNAME of the associated row in SYSIBM.LOCATIONS.	
SECURITY_OUT	An indicator specifying the DRDA security option used when DB2 SQL applications connect to any remote server associated with this TCP/IP host. Contains the following:	
	USERNAMES	Indicates whether outbound authid translation is to occur.
	IBMREQD	An indicator specifying Y if the row was supplied by IBM, or N if it was not.
	IPADDR	Contains the IP address or domain name of a remote TCP/IP host.

Figure 1: The columns of SYSIBM.IPNAMES

more other systems accessible to the local DB2 subsystem. It has one unique index, DSNFPX01, defined on the LINKNAME column. A definition of the columns in this table is shown in Figure 1.

SYSIBM.SYSDUMMY is a 'dummy' table that contains a single row. It is designed to be used in SQL statements in which a table reference is needed but the table contents are unimportant. There are no indexes on this table. It consists of a single column, IBMREQD.

THE SYSTEM CATALOG CHANGES

There are 65 new columns, 59 revised columns, 8 new indexes (all in SYSDDF), and one revised index in the DB2 V5 system catalog. The new and changed columns and indexes are provided to support the new features of DB2 V5 and to optimize performance and administration.

SIZE-RELATED CHANGES

The first major change is to the cardinality columns. Since DB2 V5 support large tablespaces, more data can be stored in a single DB2 table. The cardinality columns that existed in the DB2 catalog prior to V5 were defined as INTEGER data types. This was insufficient to store the new maximum size of large tablespaces, which can hold up to 1TB of data. Therefore, new floating point columns were added with new names. Likewise, the near-off and far-off positioning columns for indexes have been modified as well. All of the columns impacted by this change are outlined in Figure 2.

The TYPE, COLGROUPCOLNO, and NUMCOLUMNS columns were also added to both SYSCOLDIST and SYSCOLDISTSTATS to indicate what type of statistics were gathered by RUNSTATS. As of DB2 V5 there are two types of statistics that RUNSTATS can accumulate – cardinality and frequent value.

As of DB2 V5, the KEYCARD and FREQVAL parameters can be used with RUNSTATS. DB2 typically views any two columns as independent from one another. However, frequent value statistics enable DB2 to capture information about correlated columns. Columns

Table name	Old column name	New column name
SYSCOLDIST	CARD	CARDF
	FREQUENCY	FREQUENCYF
SYSCOLDISTSTATS	CARD	CARDF
	FREQUENCY	FREQUENCYF
SYSCOLUMNS	COLCARD	COLCARDF
SYSINDEXES	FIRSTKEYCARD	FIRSTKEYCARDF
	FULLKEYCARD	FULLKEYCARDF
SYSINDEXPART	FAROFFPOS	FAROFFPOSF
	NEAROFFPOS	NEAROFFPOSF
	CARD	CARDF
SYSTABLES	CARD	CARDF

Figure 2: The columns of SYSIBM.IPNAMES

are considered to be correlated with one another when their values are related in some manner. Consider, for example, CITY and STATE columns. If the CITY column is set to 'CHICAGO' it is much more common for the STATE to be set to 'IL' than any other state. However, without frequent value statistics, DB2 would consider Chicago, FL to be just as common as Chicago, IL.

Several other size-related changes were made. To accommodate large tablespaces the TYPE column was added to the SYSTABLESPACE table. If type contains the value 'L' then the tablespace is a large tablespace, instead of a regular tablespace. Large tablespaces, new to DB2 V5, can have up to 254 partitions, each containing up to 4GB.

Finally, the MAXROWS column was added to the SYSTABLESPACE table. MAXROWS indicates the maximum number of rows per page that can be stored for the tablespace. The MAXROWS parameter indicates the maximum number of rows that can be stored on a tablespace page. The default is 255, but it can range from 1 to 255.

PROCEDURE-RELATED CHANGES

Many changes were made to DB2 V5 to broaden the support provided for stored procedures. As of DB2 V5, you can use multiple Stored Procedure Address Spaces (SPAS). Doing so requires the use of the MVS Workload Manager (WLM). It allows stored procedures to be isolated in a particular address space – based on the type of processing being performed. Using multiple SPAS, you can create an environment with multiple physical address spaces for stored procedures executing at the same dispatching priority as the calling program.

Additionally, as of DB2 V5, a stored procedure can return multiple row result sets back to the calling program. If you enable result sets to be returned, stored procedures become more efficient and effective. Benefits include the following:

- Reduced network traffic, because an entire result set requires only a single network request.
- Better application design, because stored procedures do not need to loop artificially through cursors to return data one row at a time.
- Better flexibility, because more work can be done using stored procedures.

The RESULT_SETS, WLM_ENV, PGM_TYPE, EXTERNAL_SECURITY, and COMMIT_ON_RETURN columns were added to the SYSPROCEDURES table to enable features such as returning multiple result sets and Workload Manager.

DATE-RELATED CHANGES

Also, new columns were added to the system catalog to indicate when objects were created and altered. The columns ALTEREDTS and CREATEDTS contain a TIMESTAMP that indicates when the object was first created and last altered. If the object has never been altered after creation then ALTEREDTS will be the same value as CREATEDTS. These two columns have been added to the following system catalog tables: SYSDATABASE, SYSINDEXES, SYSSTOGROUP, SYSTABLESPACE, and SYSSYNONYMS (only

CREATEDTS has been added to SYSSYNONYMS because, once created, synonyms can not be altered).

Another new column, GRANTEDTS, has been introduced that is similar to CREATEDTS and ALTEREDTS. It contains a timestamp indicating when authority has been granted. This column has been added to the following system catalog tables: SYSCOLAUTH, SYSDBAUTH, SYSPLANAUTH, SYSRESAUTH, SYSTABAUTH, and SYSUSERAUTH.

Finally, columns were added to indicate bind and precompile time. The BOUNDTS column was added to the SYSPLAN table indicating the timestamp when the plan was bound. The PRECOMPTS was added to the SYSDBRM table indicating the timestamp when the DBRM was precompiled.

MISCELLANEOUS OTHER CHANGES

Many other changes were made to the system catalog tables to support DB2 Version 5. This list provides a short highlight of each change:

- ENCODING_SCHEME and several other columns were added to SYSDATABASE, SYSTABLES, and SYSTABLESPACE to enable ASCII server support and different encoding schemes. Likewise, the TRANSTYPE column in SYSSTRINGS was modified to hold more transition types.
- Check pending information, stored in a column named CHECKRID5B, was added to SYSTABLEPART and SYSTABLES.
- The KEEPDYNAMIC column was added to SYSPACKAGE and SYSPLAN to support cacheing of prepared dynamic SQL statements.
- The REOPTVAR column was added to SYSPACKAGE and SYSPLAN to indicate whether access paths are to be re-determined at execution time.
- The PIECESIZE column was added to SYSINDEXES to support the new PIECESIZE clause. PIECESIZE is used to specify the largest dataset size for a non-partitioned index.

- CREATETMTABAUTH was added to SYSUSERAUTH to control the creation of temporary tables and the TYPE column in SYSTABLES was augmented to store 'G' to specify that the table is temporary.
- The LOCKPART column was added to SYSTABLESPACE to support selective partition locking.
- The REFCOLS column was added to SYSTABAUTH and the PRIVILEGE column was added to SYSCOLAUTH to support the new REFERENCES privilege.
- The REFCOLS column was also added to the bottom of the column list for the DSNATX02 index on SYSTABUTH.
- Stored procedures can run as a main routine or a subroutine as of DB2 V5 and the PGM_TYPE column was added to SYSPROCEDURES to support this feature.
- The STATUS column was added to SYSPACKSTMT and SYSSTMT to indicate the status of the bind for each statement.
- Several new host language codes were added to the list of valid values for HOSTLANG in SYSDBRM and SYSPACKAGE. The current list of valid languages is shown in Figure 3.

Code	Programming Language
B	Assembler
C	OS/VS COBOL
D	C
F	FORTRAN
P	PL/I
2	VS COBOL II
3	IBM COBOL (Release 2 and later releases)
4	C++

Figure 3: DB2 V5 Host Language Support

- Of course, the IBMREQD column was modified by adding the dependency indicator for DB2 V5. The value 'H' indicates DB2 Version 5.
- Many other columns were modified to support new parameters, options, and features. And several other columns were deactivated (such as the old cardinality columns CARD, FULLKEYCARD, etc).

SYNOPSIS

There have been many changes to the structure of the system catalog to support DB2 Version 5. The wise DBA will study the changes and plan for the impact of these changes on their administrative functions.

Craig S Mullins
VP Operations
Platinum Technology (USA)

REXX extensions for DB2 – part 2

This month we continue the set of functions and subroutines that extend IBM REXX. These functions interface with DB2. Requests to DB2 are made under TSO using standard SQL language through the ADDRESS DB2 statement.

IRXDB2

```
$IRXDB2  START Ø
$IRXDB2  AMODE ANY
$IRXDB2  RMODE ANY
*****************************************************************
* $IRXDB2 . SET UP DB2 ENVIRONMENT.
*****************************************************************
         STM   R14,R12,12(R13) SAVE REGISTERS
         BALR  R1Ø,Ø           INIT BASE REGISTER
         USING *,R1Ø           ADRESSABILITY
         LR    R12,R13
         LA    R13,72(,R13)    NEXT SAVE AREA
```

```
         ST    R12,4(,R13)
         ST    R13,8(,R12)
         USING $IRX,R8
         USING EFPL,R1
         L     R12,EFPLARG     R12->LIST
         USING ARGUM,R12
         L     R11,EFPLEVAL    R11-> ->EVALBLOCK
         USING EVALBLOCK,R11
         L     R11,Ø(,R11)     R11-> EVALBLOCK
         B     *+4(R2)         B DEPENDING ON DESIRED FUNCTION
         B     $DB2IN          Ø: FCT $DB2INST (AND ONLY FUNCTION!)
*********************************************************************
* RC=$DB2INST('S'/'F', 'C'/'N', DESCR_VAR, VAR, ENV)
* RC IS RETURN CODE OF IBM PROG IRXSUBCM
$DB2IN   EQU   *               VARIED INITIALIZATIONS
         XC    $IRDB2DR,$IRDB2DR RESET FLAGS
*              INIT NAME OF REXX VAR CONTAINING DESCR OF COLS
         MVC   $IRCOLNV(L'$COLNT),$COLNT INIT DEFAULT NAME
         LA    R15,L'$COLNT    R15:=LNG DEFAULT NAME
         STH   R15,$IRCOLNT    STORE LNG
*              INIT NAME OF REXX VAR FOR COLS WITHOUT NAME
         MVC   $IRCOLV(L'$COL),$COL INIT DEFAULT NAME
         LA    R15,L'$COL      R15:=LNG DEFAULT NAME
         STH   R15,$IRCOL      STORE LNG
         MVC   $IRDB2EN+8(8),MODULE STORE NAME OF MODULE
*              SET UP ENV FOR ADDRESS DB2:
         MVI   $IRDB2EN,C' '   1ST B. OF ENTRY OF HOST COMMAND ENV TABL
MVC   $IRDB2EN+1(L'$IRDB2EN-1),$IRDB2EN BLANK THE ENTRY
         MVC   $IRDB2EN(L'DB2),DB2 STORE "DB2"
         MVC   $IRDB2EN+8(8),MODULE STOCRE NAME OF MODULE
         CLI   $IRNBARG,5      AT LEAST 5 ARG ?
         BL    $DB2IN12        NO, B
         L     R15,ARGUM5P     YES. R15->5TH ARG
         L     R1,ARGUM5L      R1:=LNG ARG
         LTR   R1,R1           LNG=Ø ?
         BNP   $DB2IN12        YES, DON'T CARE OF IT
         CH    R1,=H'8'        LNG ARG > 8 ?
         BNH   $DB2IN8         NO, B
         LA    R1,8            YES. TRUNCATE IT TO 8$DB2IN8  EQU   *
         MVC   $IRDB2EN(8),BLANCS BLANK THE ZONE
         BCTR  R1,Ø            LNG-1 FOR EX
         EX    R1,$DB2INM1     STORE NAME FOR "ADDRESS NAME"
         MVC   $IRZONE(L'$IRDB2EN),$IRDB2EN STORE IN WORK AREA
*              FOR THE QUERY (WILL GET RUN OVER IF NON-EXISTENT)
$DB2IN12 EQU   *               CREATE PARAM FOR IRXSUBCM
         LA    R15,QUERY       1ST PARAM: "QUERY   "
         ST    R15,$IRMFE      STORE ADDR OF 1ST PARAM
         LA    R15,$IRZONE     2ND PARAM: ADDR TABLE ENTRY
         ST    R15,$IRFØ       STORE ADDRESS
         LA    R15,$IRFØ       R15->ADDR TABLE ENTRY
         ST    R15,$IRMFE+4    STORE ADDR 2ND PARAM
```

```
         LA    R15,LENT        3RD PARAM: LNG OF 1 ENTRY
         ST    R15,$IRMFE+8    STORE ADDR 3RD PARAM
         LA    R15,$IRZONE     4TH PARAM: NAME OF ENVIRONMENT
         ST    R15,$IRMFE+12   STORE ADDR 4TH PARAM
         OI    $IRMFE+12,X'8Ø' 1ST BIT TO 1: END OF PARAMETERS
         L     RØ,$IRXENVB     RØ->ENV BLOCK
         LA    R1,$IRMFE       R1->LIST OF PARAM
         LINK  EP=IRXSUBCM     CALL REXX
         CH    R15,=H'8'       THIS ENTRY WAS REALLY NON-EXISTENT?
BNE   $DB2IN16        NO, B (ALREADY EXISTS, OR OTHER ERROR)
*                              ENTRY WAS NON-EXISTENT. CREATE IT
         LA    R15,ADD         1ST PARAM: "ADD     "
         ST    R15,$IRMFE      STORE ADDR 1ST PARAM
         LA    R15,$IRDB2EN    2ND PARAM: ADDR TABLE ENTRY
         ST    R15,$IRFØ       STORE ADDR
         LA    R15,$IRDB2EN    4TH PARAM: NAME OF ENVIRONMENT
         ST    R15,$IRMFE+12   STORE ADDR 4TH PARAM
         OI    $IRMFE+12,X'8Ø' 1ST BIT TO 1: END OF PARAM
         L     RØ,$IRXENVB     RØ->ENV BLOCK
         LA    R1,$IRMFE       R1->LIST OF  PARAM
         LINK  EP=IRXSUBCM     CALL REXX TO CREATE A NEW ENVIRONMENT
$DB2IN16 EQU   *
         CVD   R15,$IRPACK1    CONV RETURN CODE TO DECIMAL
         UNPK  EVALBLOCK_EVDATA(16),$IRPACK1 CONV IN EXTENDEC
         LA    R14,16          LNG OF RESULT
         ST    R14,EVALBLOCK_EVLEN STORE LNG OF RESULT
*                              PROCESS THE OTHER ARGUMENTS
         CLI   $IRNBARG,1      AT LEAST 1 ARG ?
         BL    $DB2IN9Ø        NO, DON'T TOUCH ANYTHING
* 1ST ARG: 'S': STANDARD NAMES FOR COLUMNS
*          'F': FORCE NAME OF COLS WITH 4TH ARGUMENT
         L     R15,ARGUM1P     R15->STRING
         L     R14,ARGUM1L     R14:=LNG OF ARG
         LTR   R14,R14         LNG ARG =Ø ?
         BNP   $DB2IN2Ø        YES, THIS IS NOT AN INTERESTING ARG
         CLI   Ø(R15),C'S'     'S' (DEFAULT VALUE) ?
         BE    $DB2IN2Ø        YES, DON'T DO ANYTHING
         CLI   Ø(R15),C'F'     'F' (FORCE) ?
         BNE   ERR5Ø           NO, B ERROR
         OI    $IRDB2DR,$IRDB2FO YES. INDICATE "FORCE NAME"
$DB2IN2Ø EQU   *
         CLI   $IRNBARG,2      AT LEAST 2 ARG ?
         BL    $DB2IN9Ø        NO, DON'T TOUCH ANYTHING
* 2ND ARG: 'C': STD: CONVERT DATA COMING FROM DB2
*          'N': DON'T CONVERT
         L     R15,ARGUM2P     R15->2ND ARG
         L     R14,ARGUM2L     R14.=LNG 2ND ARG
         LTR   R14,R14         LNG ARG = Ø ?
         BNP   $DB2IN3Ø        YES, NOTHING TO DO
         CLI   Ø(R15),C'C'     CONVERSION REQUIRED (DEFAULT VALUE) ?
         BE    $DB2IN3Ø        YES, NOTHING TO DO
```

```
         CLI   Ø(R15),C'N'     NO CONVERSION ?
         BNE   ERR5Ø           NO, B ERROR
         OI    $IRDB2DR,$IRDB2NC YES. INDICATE "NO CONVERSION"
$DB2IN3Ø EQU   *
         CLI   $IRNBARG,3      AT LEAST 3 ARG ?
         BL    $DB2IN9Ø        NO, NOTHING TO DO
* 3RD ARGUMENT: NAME OF REXX VAR TO CONTAIN DESCR OF COLUMNS
         L     R15,ARGUM3P     R15->3RD ARG
         L     R14,ARGUM3L     R14:=LNG 3RD ARG
         LTR   R15,R15         DUMMY ARGUMENT ?
         BZ    $DB2IN4Ø        YES, NOTHING TO DO
         LA    RØ,$IRCOLVL     RØ:=MAX LNG OF NAME
         CR    R14,RØ          LNG TOO HIGH ?
         BNH   $DB2IN35        NO, B
         LA    R14,$IRCOLVL    YES, TRUNCATE IT
$DB2IN35 EQU   *
         STH   R14,$IRCOLNT    STORE LNG PREFIX OF VAR NAME
         LTR   R14,R14         LNG=Ø ?
         BNP   $DB2IN4Ø        YES, DON'T STORE THE NAME
         BCTR  R14,Ø           LNG-1 FOR EX
         EX    R14,$DB2INLM    STORE PREFIX OF VAR NAME
$DB2IN4Ø EQU   *
         CLI   $IRNBARG,4      AT LEAST 4 ARG ?
         BL    $DB2IN9Ø        NO, NOTHING TO DO
* 4TH ARGUMENT: NAME OF REXX VAR FOR COLUMNS WITHOUT A NAME
         L     R15,ARGUM4P     R15->4TH ARG
         L     R14,ARGUM4L     R14.=LNG 4TH ARG
         LTR   R15,R15         DUMMY ARGUMENT ?
         BZ    $DB2IN58        YES, NOTHING TO DO
         LA    RØ,$IRCOLVL     RØ:=MAX LNG OF NAME
         CR    R14,RØ          LNG TOO HIGH ?
         BNH   $DB2IN55        NO, B
         LA    R14,$IRCOLVL    YES, TRUNCATE IT
$DB2IN55 EQU   *
         STH   R14,$IRCOL      STORE LNG OF PREFIX FOR VAR NAME
         LTR   R14,R14         LNG=Ø ?
         BNP   $DB2IN58        YES, DON'T STORE THE NAME
         BCTR  R14,Ø           LNG-1 FOR EX
         EX    R14,$DB2INLN    STORE PREFIX OF VAR NAME
$DB2IN58 EQU   *
$DB2IN9Ø EQU   *
         CLI   $IRNBARG,5      MORE THAN 5 ARGUMENTS ?
         BH    ERR15           YES, B ERROR
* (RETURN CODE IN EVALDATA HAS ALREADY BEEN SET)
         B     FIN             END WITH NORMALIZATION
$DB2INLM MVC   $IRCOLNV(R14-R14),Ø(R15)
$DB2INLN MVC   $IRCOLV(R14-R14),Ø(R15)
$DB2INM1 MVC   $IRDB2EN(R1-R1),Ø(R15)
*********************************************************************
FIN      EQU   *               END FOR NUMERIC
         LA    R1,4            R1:=4: NUMERIC TO BE NORMALIZED
```

```
         B     FIN2
FINCHN   EQU   *               END FOR STRING, OR NUM NOT TO BE NORM.
         LA    R1,Ø            R1:=Ø: STRING
FIN2     EQU   *
         L     R13,4(,R13)
         L     R14,12(,R13)    RESTORE R14
         L     RØ,2Ø(,R13)     RESTORE RØ
         LM    R2,R12,28(R13)  RESTORE REGS BUT R1
         XR    R15,R15         RETURN CODE Ø
         BR    R14
*********************************************************************
ERR15    LA    R15,ERØØ7       ERROR NUMBER OF ARG
         B     ERR
ERR5Ø    LA    R15,ERØØ8       ERROR ARG FCT $DB2INST
         B     ERR
ERR      EQU   *
         L     R13,4(,R13)
         L     R14,12(,R13)    RESTORE R14
         LM    RØ,R12,2Ø(R13)  RESTORE REG BUT R15
         BR    R14
*DATA**************************************************************
         LTORG
ADD      DC    CL8'ADD'
QUERY    DC    CL8'QUERY'
DB2      DC    C'DB2'          FOR "ADDRESS DB2"
LENT     DC    F'32'           LNG OF 1 ENTRY IN TABLE
MODULE   DC    CL8'$IRXDB2H'
$COLNT   DC    C'$COLNT.'      NAME OF REXX VAR FOR DESCR OF COLS
$COL     DC    C'$COL'         NOAME OF REXX VAR FOR COLS W/OUT NAME
BLANCS   DC    CL8' '
*DSECTS************************************************************
         COPY  $IRXDSEC
         IRXEFPL
         IRXEVALB
         END
```

IRXDB2H

```
$IRXDB2H CSECT
$IRXDB2H AMODE ANY
$IRXDB2H RMODE 24              BECAUSE OF OS/VS COBOL $IRXCNV2
         EXTRN DSNALI
NBCURMAX EQU   C'2'            2 CURSORS MAXI
NBCURMAB EQU   2               MUST BE = NBCURMAX, IN BINARY
NBCUHMAX EQU   C'2'            2 CURSORS "WITH HOLD" MAXI
* CURSORS ARE C1, C2, H1, H2, (NUMBERED Ø, 1, 2, 3)
*****************************************************************
* $IRXDB2H. DB2 INTERFACE. AUTHOR: PATRICK LELOUP.
* CAUTION, THIS PROG IS CALLED BY "ADDRESS XX", AND IS NOT CALLED
* BY IRXFLOC, LIKE OTHER FUNCTIONS. IT MUST ALLOCATE REGISTERS
```

```
* SAVE AREA BY ITSELF.
*****************************************************************
         STM   R14,R12,12(R13) SAVE REGISTERS
         BASR  R11,Ø           INIT BASE REGISTER
         USING *,R11,R12
         LA    R12,4Ø95(,R11)  INIT 2ND BASE ...
         LA    R12,1(,R12)     ..REGISTER
         LR    R9,R1           R9->PARM AT ENTRY
         LR    R6,RØ           R6->ENV BLOCK
         GETMAIN RU,LV=SAUVL,LOC=BELOW BELOW FOR $IRXCNV2
         ST    R1,8(,R13)
         ST    R13,4(,R1)
         LR    R13,R1
         USING ENVBLOCK,R6
         L     R6,ENVBLOCK_WORKBLOK_EXT R6->WORK BLOCK EXTENSION
         DROP  R6
         USING WORKBLOK_EXT,R6
         L     R8,WORKEXT_USERFIELD     R8->USER FIELD
         DROP  R6
         USING $IRX,R8
         LTR   R8,R8           OUR TABLE EXISTS?
         BZ    ERR54           NO, B ERR (FCT $DB2INST NOT CALLED)
         USING $IRX,R8
         LR    R1,R9           R1->PARAM AT ENTRY
         L     R7,4(,R1)       R7->2ND PARM (STRING)
         L     R7,Ø(,R7)       R7->STRING FOR DB2
         ST    R7,$IRF1        STORE BEGINNING ADDR OF STRING
         L     R6,8(,R1)       R6->3RD PARM (LNG STRING)
         L     R6,Ø(,R6)       HERE, R6=LNG OF STRING, R7->STRING
         L     R1Ø,$IRXDB2A    R1Ø->WORK AREA FOR DB2 (CONTAINS SQLCA)
         LTR   R1Ø,R1Ø         DOES THE ZONE EXIST?
         BNZ   ZONE1Ø          YES, B
         GETMAIN RU,LV=DB2L,LOC=ANY NO. ACQUIRE IT
         ST    R1,$IRXDB2A     STORE ADDR OF WORK AREA
         LR    R1Ø,R1          R1Ø->WORK AREA
         LA    R15,DB2L        R15:=LNG OF AREA
         ST    R15,$IRXDB2L    STORE LNG OF AREA
ZONE1Ø   EQU   *
         USING DB2D,R1Ø
         LA    R9,SQLDSEC2     R9->SQLDSECT
         USING SQLDSECT,R9
         LA    R2,TABFCT       R2->TABLE OF KEYWORDS
         USING FONCTD,R2
FCT1Ø    EQU   *               GET CODE OF REQUESTED FUNCTION
         LH    R1,FONCTL       R1:=LNG OF THIS FUNCTION
         CR    R1,R6           STRING LONG ENOUGH FOR THIS FCT?
         BH    FCT2Ø           NO, LET'S SEE THE NEXT ONE
         BCTR  R1,Ø            YES. LNG-1 FOR EX
         EX    R1,FCTCLC       IS IT THIS FUNCTION ?
         BE    FCT4Ø           YES, WE HAVE FOUND THE FUNCTION
FCT2Ø    EQU   *
```

```
         LA    R2,FONCTDL(,R2) NEXT ENTRY
         CLI   Ø(R2),X'FF'     END OF TABLE ?
         BNE   FCT1Ø           NO, LOOP
         B     ERR45           FUNCTION NOT FOUND. B SYNTAX ERROR
FCT4Ø    EQU   *               WE HAVE FOUND THE DESIRED FUNCTION
         LA    R1,1(,R1)       RESTORE LNG OF FUNCTION
         LA    R7,Ø(R1,R7)     R7->BEGIN. OF PARM (OR BLANKS)
         SR    R6,R1           R6:=REMAINING LENGTH
         AR    R6,R7           R7->1ST BYTE AFTER STRING
         L     R15,FONCTA      R15->PROCESSING MODULE
         BR    R15             BRANCH TO  MODULE (W/ R1=LNG-1 OF FUNCT)
* WE CALL MODULE WITH R7->PARAMETERS (CAN BEGIN WITH BLANKS)
*                     R6->END OF PARAMETERS+1
*                     $IRF1->BEGINNING OF STRING (FUNCTION INCLUDED)
FCTCLC   CLC   FONCT(R1-R1),Ø(R7)
         DROP  R2
*********************************************************************
         DSNDDECP
         EXEC  SQL DECLARE C1 CURSOR FOR STMT1
         EXEC  SQL DECLARE STMT1 STATEMENT
         EXEC  SQL DECLARE C2 CURSOR FOR STMT2
         EXEC  SQL DECLARE STMT2 STATEMENT
         EXEC  SQL DECLARE H1 CURSOR WITH HOLD FOR STMTH1
         EXEC  SQL DECLARE STMTH1 STATEMENT
         EXEC  SQL DECLARE H2 CURSOR WITH HOLD FOR STMTH2
         EXEC  SQL DECLARE STMTH2 STATEMENT
*********************************************************************
CONNECT  EQU   *               CONNECT NAME_OF_DB2
         BAS   R14,RECHNB      SEARCH FOR 1ST NON-BLANK
         LTR   R7,R7           SOMETHING FOUND?
         BZ    ERR45           NO, B SYNTAX ERROR
*                              STORE DB2ID INTO $IRDB2
         MVC   $IRDB2,BLANCS   BLANK IT
         LA    R1,$IRDB2       R1->RECEIVE AREA
         LA    RØ,8            8 BYTES MAXI
CONNEC1Ø EQU   *
         MVC   Ø(1,R1),Ø(R7)   STORE 1 BYTE
         LA    R1,1(,R1)       NEXT BYTE IN RECEIVING AREA
         LA    R7,1(,R7)       NEXT BYTE IN SENDING AREA
         CR    R7,R6           BEYOND LIMIT?
         BNL   CONNEC2Ø        YES, STOP IT
         BCT   RØ,CONNEC1Ø     NO, LOOP
CONNEC2Ø EQU   *
         LA    R15,LOPEN       R15->1ST PARM ("OPEN")
         ST    R15,$IRMFE      STORE ADDR OF 1SR PARM
         LA    R15,$IRDB2      R15->2ND PARM (DB2ID)
         ST    R15,$IRMFE+4    STORE ADDR OF 2ND PARM
         LA    R15,PLAN        R15->3RD PARM (NAME OF PLAN)
         ST    R15,$IRMFE+8    STORE ADDR OF 3RD PARM
         OI    $IRMFE+8,X'8Ø'  INDICATE LAST PARM
         LA    R1,$IRMFE       R1->LIST OF PARAMETERS
```

```
         L     R15,VDSNALI     R15->DSNALI
         BASR  R14,R15
         ST    RØ,$IRDB2RE     STORE REASON CODE
         ST    R15,$IRDB2RT    STORE RETURN CODE
         B     STOCREAS
*********************************************************************
DISCONN  EQU   *               DISCONN <SYNC>!<ABRT>
         LA    R15,LCLOSE      R15->1ST PARAM ("CLOSE")
         ST    R15,$IRMFE      STORE ADDR 1ST PARAM
         LA    R15,SYNC        R15->2ND PARAM ("SYNC" BY DEFAULT)
         ST    R15,$IRMFE+4    STOCKAGE ADR 2ND PARAM
         BAS   R14,RECHNB      SEARCH FOR 1ST NON BLANK
         LTR   R7,R7           SOMETHING FOUND?
         BZ    DISCON2Ø        NO, B SYNC BY DEFAULT
         ST    R7,$IRMFE+4     STORE ADDR 2ND PARAM
DISCON2Ø EQU   *
         OI    $IRMFE+4,X'8Ø'  INDICATE LAST PARAM
         LA    R1,$IRMFE       R1->LIST OF PARAMETERS
         L     R15,VDSNALI     R15->DSNALI
         BASR  R14,R15
         ST    RØ,$IRDB2RE     STORE REASON CODE
         ST    R15,$IRDB2RT    STORE RETURN CODE
         B     STOCREAS
*********************************************************************
DECLARE  EQU   *               DECLARE Cx/Hx CURSOR <WITH HOLD> FOR...
         L     R7,$IRF1        R7->BEGINNING OF STRING
         LR    R1,R6           R1:=...
         SR    R1,R7           LNG OF DATA
         LA    RØ,DECURL       MIN L FOR DECLARE CURSOR W/O "WITH HOLD"
         CR    R1,RØ           LNG<=MIN LNG ?
         BNH   ERR45           YES, B SYNTAX ERROR
         CLC   DECUR1,Ø(R7)    "DECLARE C" ?
         BE    DECL3           YES, B
         CLC   DECURH1,Ø(R7)   "DECLARE H" ?
         BNE   ERR45           NO, B SYNTAX ERROR
*                              "DECLARE H9 CURSOR WITH HOLD FOR "
         LA    RØ,DECURHL      RØ:=MIN LNG OR REQUEST
         CR    R1,RØ           LNG<=MIN LNG ?
         BNH   ERR45           YES, B SYNTAX ERROR
         LA    R2,L'DECURH1+L'DECURH2(,R7) R2->ZONE BEHIND "DECLARE HX"
         CLC   DECURH3,Ø(R2)   " CURSOR WITH HOLD FOR " ?
         BNE   ERR45           NO, B SYNTAX ERROR
         LA    R2,L'DECURH1(,R7) R2->CURSOR NUMBER
         CLI   Ø(R2),C'1'      THIS NUMBER NUMERIC?
         BL    ERR45           NO, B SYNTAX ERROR
         CLI   Ø(R2),NBCUHMAX  CURSOR NUMBER TOO HIGH ?
         BH    ERR45           YES, B SYNTAX ERROR
         IC    R2,Ø(,R2)       R2:=CURSOR NUMBER, EXTENDED
         N     R2,FØ           R2:=CURSOR NUMBER, IN BINARY
         BCTR  R2,Ø            CURSOR NUMBER / Ø
         LA    RØ,NBCURMAB     RØ:=HIGHEST CURSOR NUMBER W/O HOLD
```

```
         AR    R2,RØ           CURSOR NUMBER / Ø, "C" CURSORS INCL.
         STH   R2,$IRH1        STORE CURSOR NUMBER
         LA    RØ,DECURHL      RØ:=LNG WORDS IN FRONT OF "SELECT"
         B     DECL4
DECL3    EQU   *               "DECLARE C9 CURSOR FOR "
         LA    RØ,DECURL       RØ:=MIN LNG OF REQUEST
         CR    R1,RØ           LNG<=MIN LNG ?
         BNH   ERR45           YES, B SYNTAX ERROR
         LA    R2,L'DECUR1+L'DECUR2(,R7) R2->ZONE BEHIND "DECLARE CX"
         CLC   DECUR3,Ø(R2)    " CURSOR FOR " ?
         BNE   ERR45           NO, B SYNTAX ERROR
         LA    R2,L'DECUR1(,R7) R2->CURSOR NUMBER
         CLI   Ø(R2),C'1'      NUMBER IS NUMERIC?
         BL    ERR45           NO, B SYNTAX ERROR
         CLI   Ø(R2),NBCURMAX  CURSOR NUMBER TOO HIGH?
         BH    ERR45           YES, B SYNTAX ERROR
         IC    R2,Ø(,R2)       R2:=CURSOR NUMBER, EXTENDED
         N     R2,FØ           R2:=CURSOR NUMBER IN BINARY
         BCTR  R2,Ø            CURSOR NUMBER / Ø
         STH   R2,$IRH1        STORE CURSOR NUMBER
         LA    RØ,DECURL       RØ:=LNG WORDS IN FRONT OF "SELECT"
DECL4    EQU   *
         AR    RØ,R7           RØ->BEGINNING OF SELECT (WE HOPE...)
         ST    RØ,$IRF1        STORE BEGINNING ADDR
         BAS   R14,STOSQL      STORE INTO DB2SQL
*                               GET SQLDA FOR THIS CURSOR
         LH    R4,$IRH1        R4:=CURSOR NUMBER, BIN / Ø
         MH    R4,=AL2($IRSQLDL) * NB BYTES IN 1 ENTRY OF ADDR/LNG
         LA    R15,$IRSQLP1    R15->1ST ENTRY
         AR    R4,R15          R4->OUR ENTRY
         USING $IRSQLAD,R4
         L     R5,$IRSQLAØ     R5->SQLDA FOR THIS CURSOR
         USING SQLDA,R5
         LTR   R5,R5           SQLDA ALREADY GOT?
         BNZ   DECL5           YES, B
         L     RØ,LMAXSQL      RØ:=LNG FOR A BIG SQLDA
         GETMAIN RU,LV=(Ø),LOC=ANY GETMAIN OUR SQLDA
         ST    R1,$IRSQLAØ     STORE ADDR OF SQLDA
         ST    RØ,$IRSQLLØ     STORE LNG OF SQLDA
         LR    R5,R1           R5->OUR SQLDA
         MVC   SQLDAID,IDSQLDA  "SQLDA"
         ST    RØ,SQLDABC      STORE LNG OF SQLDA INTO SQLDA
         LA    R15,NBSQLVAR    R15:=NB OF SQLVAR FOR THIS SQLDA
         STH   R15,SQLN        STORE INTO SQLDA
DECL5    EQU   *               EXEC SQL PREPARE FOR THE PROPER CURSOR
         LH    R1,$IRH1        R1:=CURSOR NUMBER, BIN / Ø
         SLL   R1,2            *4
         B     *+4(R1)         B ACCORDING TO CURSOR NUMBER
         B     DECLC1          B IF CURSOR C1
         B     DECLC2          B IF CURSOR C2
         B     DECLH1          B IF CURSOR H1
```

```
         B     DECLH2          B IF CURSOR H2
DECLC1   EQU   *
         EXEC SQL PREPARE STMT1 INTO :SQLDA FROM :DB2SQL
         B     DECL2Ø
DECLC2   EQU   *
         EXEC SQL PREPARE STMT2 INTO :SQLDA FROM :DB2SQL
         B     DECL2Ø
DECLH1   EQU   *
         EXEC SQL PREPARE STMTH1 INTO :SQLDA FROM :DB2SQL
         B     DECL2Ø
DECLH2   EQU   *
         EXEC SQL PREPARE STMTH2 INTO :SQLDA FROM :DB2SQL
         B     DECL2Ø
DECL2Ø   EQU   *
         LTR   R15,R15         RETURN CODE OK ?
         BNZ   ERR49           NO, B ERROR
         L     R15,SQLCODE     R15:=SQLCODE
         LTR   R15,R15         WRONG RETURN CODE?
         BNZ   FINEXEC         YES, B
*                              ALLOCATE RECEIVING AREA FOR COLUMNS
         L     R1,$IRCOLAØ     R1->VALUES AREA
         LTR   R1,R1           ALREADY GETMAINED?
         BZ    DECL22          NO, B
         L     RØ,$IRCOLLØ     YES. R3:=LNG OF THIS AREA
         FREEMAIN RU,LV=(Ø),A=(1) FREE THIS AREA
         XR    RØ,RØ           ZERO RØ FOR ...
         ST    RØ,$IRCOLAØ     ZERO ADDR OF AREA
         ST    RØ,$IRCOLLØ     ZERO LNG AREA
DECL22   EQU   *
         LH    RØ,SQLD         RØ:=NUMBER OF COLUMNS IN SELECT
         LTR   RØ,RØ           Ø COLUMN RECEIVED?
         BNP   DECL7Ø          YES, B NOTHING TO DO
         LA    R3,SQLVAR       R3->1ST SQLVAR
         USING SQLVARN,R3
         XR    R2,R2           R2 WILL CONTAIN LNG AREA TO ACQUIRE
         MVC   $IRH1,SQLTYPE   STORE DATA TYPE
         NI    $IRH1+1,X'FE'   ZERO BIT 'MAY BE NULL'
DECL25   EQU   *
         LH    R15,SQLLEN      R15:=LNG OF THIS COLUMN
         CLC   $IRH1,T484      DECIMAL?
         BNE   DECL4Ø          NO, B
         XR    R15,R15         YES. R15:=...
         IC    R15,SQLLEN      LNG OF THIS COLUMN
         LA    R15,2(,R15)     +2 FOR THE SIGN
         SRA   R15,1           /2: RIGHT LNG IN BYTES
DECL4Ø   EQU   *
         CLC   $IRH1,T464      GRAPHIC ?
         BE    DECL41          YES, B
         CLC   $IRH1,T472      GRAPHIC ?
         BNE   DECL42          NO, B
DECL41   EQU   *               GRAPHIC
```

```
         SLA   R15,1           LNG * 2
DECL42   EQU   *
         AR    R2,R15          UPDATE LNG AREA TO ACQUIRE
         LA    R2,4(,R2)       +2 FOR NULL +2 FOR LNG
         LA    R3,SQLSIZV(,R3) NEXT SQLVAR
         BCT   RØ,DECL25       LOOP FOR ALL SQLVAR
         GETMAIN RU,LV=(R2),LOC=ANY
         ST    R1,$IRCOLAØ     STORE ADDRESS OF AREA
         ST    R2,$IRCOLLØ     STORE LNG OF AREA
*                              STORE VARIOUS SUB-AREA ADDR. IN SQLDA
         LH    RØ,SQLD         RØ:=NUMBER OF COLUMNS
         LTR   RØ,RØ           Ø COL ?
         BNP   DECL7Ø          YES, NOTHING TO DO
         LA    R3,SQLVAR       R3->1ST SQLVAR
DECL5Ø   EQU   *
         ST    R1,SQLIND       STORE ADDR FOR NULL INDICATOR
         LA    R1,2(,R1)       R1->LNG IN THIS SUB-AREA
         CLC   $IRH1,T448      TYPE VARYING ?
         BE    DECL6Ø          YES, B
         CLC   $IRH1,T456      TYPE VARYING ?
         BE    DECL6Ø          YES, B
         CLC   $IRH1,T464      TYPE VARYING ?
         BE    DECL6Ø          YES, B
         CLC   $IRH1,T472      TYPE VARYING ?
         BE    DECL6Ø          YES, B
         LA    R1,2(,R1)       NON-VARYING. +2 ON RECEIVING AREA
DECL6Ø   EQU   *
         ST    R1,SQLDATA      STORE RECEIVING ADDR OF VALUE
         LH    R15,SQLLEN      R15:=LNG OF COL
         CLC   $IRH1,T484      PACKED DECIMAL ?
         BNE   DECL65          NO, B
         XR    R15,R15         YES.
         IC    R15,SQLLEN      R15:=PRECISION OF DECIMAL
         LA    R15,2(,R15)     +2 FOR THE SIGN
         SRA   R15,1           /2 : RIGHT LNG IN BYTES
DECL65   EQU   *
         AR    R1,R15          UPDATE PTR IN OUTPUT AREA
         LA    R3,SQLSIZV(,R3) NEXT SQLVAR
         BCT   RØ,DECL5Ø       LOOP FOR ALL SQLVAR
DECL7Ø   EQU   *               SQLDA IS NOW READY FOR A FETCH
* ASSIGNMENT OF VARIABLES $COLN.I CONTAINING NAME COL+TYPE
         LH    R15,$IRCOLNT    R15:=LNG PREFIX OF REXX VAR NAME
         LTR   R15,R15         NO VAR ?
         BNP   DECL9Ø          NO VAR. DON'T FILL ANYTHING
         XR    R15,R15         R15:=CURRENT COL NUMBER
         STH   R15,$IRHØ       SAVE IT for ANONYM. COLS, AND FOR TOTAL
         LH    RØ,SQLD         RØ:=NUMBER OF COLS
         LTR   RØ,RØ           Ø COL ?
         BNP   DECL8Ø          YES, NOTHING TO DO (BUT $COLN.Ø)
         LA    R3,SQLVAR       R3->1ST SQLVAR
DECL75   EQU   *
```

```
         LH    R15,$IRHØ       R15:=CURRENT COL NUMBER
         LA    R15,1(,R15)     +1
         STH   R15,$IRHØ       SAVE IT
         MVC   $IRH1,SQLTYPE   STORE COLUMN TYPE
         NI    $IRH1+1,X'FE'   ZERO BIT 'MAY BE NULL'
         LA    R14,SQLNAME+2   R14->BEGINNING OF POSSIBLE COL NAME
         LH    R15,SQLNAME     R15:=LNG OF COL NAME
         LTR   R15,R15         THIS COL HAS A NAME?
         BP    DECL752         YES, B
DECL751  EQU   *               NO NAME FOR THIS COL.
         LA    R14,COLSANOM    R14->INDICATOR FOR ANONYMOUS COL
         LA    R15,L'COLSANOM  R15:=LNG OF THIS INDICATOR
DECL752  EQU   *               R14->NAME OF THE COLUMN
         BCTR  R15,Ø           LNG-1 FOR EX
         EX    R15,DECLMVC2    STORE COL NAME
         LA    R1,DB2SQLS+1    R1->BEGINNING OF COL'S NAME + 1
         AR    R1,R15          R1->1ST BYTE AFTER COL'S NAME
DECL755  EQU   * HERE,R1->1ST BYTE AFTER COL'S NAME IN OUTPUT AREA
         LA    R15,TYPES       R15->LIST OF TYPES
         USING TYPED,R15
DECL76   EQU   *               WE ARE NOW SEARCHING FOR TYPE
         CLI   TYPET,X'FF'     END OF TABLE ?
         BE    DECL77          YES, THIS TYPE NOT FOUND
         CLC   $IRH1,TYPET     NO. IS IT THIS TYPE?
         BE    DECL77          YES, B
         LA    R15,TYPEDL(,R15) NO, NEXT TYPE
         B     DECL76          LOOP
DECL77   EQU   *               WE ARE ON THE PROPER TYPE(OR UNKNOWN!)
         LH    R2,TYPEL        R2:=LNG-1 OF NAME OF THIS TYPE
         L     R15,TYPEA       R15->NAME FOR THIS TYPE
         DROP  R15
         EX    R2,DECLMVC      STORE NAME OT THIS TYPE
         LA    R1,Ø(R1,R2)     R1->LAST BYTE OF NAME
         CLI   Ø(R1),C'('      ENDS WITH ( ?
         BNE   DECL78          NO, B
*                     NAME ENDS WITH (: WE MUST STORE THE LENGTH TOO
         LA    R1,1(,R1)       R1->1ST BYTE AFTER "("
         LH    R15,SQLLEN      R15:=LNG OF COL, OR NEARLY
         CLC   $IRH1,T48Ø      IS IT FLOATING ?
         BNE   DECL775         NO, B
         CH    R15,=H'4'       IS IT SINGLE FLOATING ?
         BNE   DECL772         NO, B
         MVC   Ø(2,R1),=CL2'21' YES. STORE PRECISION
         LA    R1,2(,R1)       UPDATE OUTPUT POINTER
         B     DECL779
```

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

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

DB2 bufferpool maintenance

Over the years, many DB2 bufferpool maintenance routines have been written and made available via a variety of sources. What I have never found is a simple on-line routine that gives you a full picture of the DB2 bufferpools. The only way was to wade through a DISPLAY BUFFERPOOL listing with a pencil and a piece of paper. This REXX is my attempt at providing that solution.

It is not as exhaustive as many proprietary products because it does not provide reporting facilities etc, but, then again, it costs far less, and, with a bit of forward thinking and knowledge of DB2, provides the DBA with all the information required to monitor and/or amend the bufferpools in any given subsystem. I have certainly never needed to probe more deeply than using this REXX together with a knowledge of the workload on the given system.

The REXX is basically a 'DO FOREVER' REXX loop that allows the operator to monitor and/or amend active and/or allocated bufferpools *ad infinitum*. The basic panel gives the operator the ability to select the DB2 subsystem, active or allocated bufferpools, and incremental or cumulative statistics. From this the operator can continue to monitor the bufferpools, by pressing enter, or ask for further details by 'Selecting' or ALTER the bufferpool by 'Amending' a given bufferpool.

The REXX gives two vital pieces of information – the VP read efficiency, and the hiperpool READ/WRITE ratio. The VP read efficiency is basically the DB2 system hit ratio expressed as a percentage. The hiperpool READ/WRITE ratio is exactly what it says, also expressed as a percentage.

REXX

```
/***************************** REXX ********************************/
/*          BUFFPOOL - DB2 BUFFERPOOL MAINTENANCE                  */
/*   THIS REXX ALLOWS THE USER TO INTERROGATE THE BUFFERPOOLS IN */
/*   USE OR DEFINED TO A GIVEN DB2 SUBSYSTEM.                      */
/*                                                                 */
/*   SCREEN 1 (BUFF001P) LETS THE USER INSERT WHICH DB2            */
/*   SUBSYSTEM IS REQUIRED, WHICH TYPE OF BUFFERPOOLS TO LOOK AT */
```

```
/*   AND WHICH TIME SCALE TO TAKE.                                    */
/*                                                                    */
/*   THIS SCREEN IS CONSTANTLY REFRESHABLE BY SIMPLY HITTING          */
/*   <ENTER> OR BY CHANGING THE BUFFERPOOL AND/OR SUBSYSTEM           */
/*   CRITERIA AND HITTING <ENTER>.                                    */
/*                                                                    */
/*   THE 'BUFFERPOOL EFFICIENCY' (VP READ EFF)                        */
/*        IS TAKEN AS FOLLOWS :-                                      */
/*   (GETPGS - ASYNC - SYNCIO)/GETPGS                                 */
/*   WHERE GETPGS = SUM OF RANDOM AND SEQUENTIAL GETPAGES             */
/*                  FROM THE DSNB411 MESSAGE.                         */
/*         ASYNC  = SUM OF THE SEQUENTIAL, LIST AND DYNAMIC           */
/*                  PREFETCH ASYNC I/OS IN THE DSNB412, 413 &         */
/*                  414 MESSAGES RESPECTIVELY.                        */
/*         SYNCIO = SUM OF THE RANDOM AND SEQUENTIAL SYNCIOS          */
/*                  FROM THE DSNB411 MESSAGE.                         */
/*         EXPRESSED AS A PERCENTAGE.                                 */
/*                                                                    */
/*   FROM THIS SCREEN ANY ONE OF THE BUFFERPOOLS SHOWN ARE            */
/*   'SELECTABLE' EITHER FOR MORE INFORMATION OR TO ALTER ONE OR       */
/*   MORE OF THE AMENDABLE VALUES.                                    */
/*                                                                    */
/*   IF A BUFFERPOOL IS SELECTED,  IE AN 'S' IS INSERTED, THE         */
/*   DETAILS SCREEN (BUFF003P) IS SHOWN. NO AMENDMENTS OR             */
/*   SELECTIONS ARE AVAILABLE FROM THIS SCREEN, IT IS                 */
/*   DISPLAY ONLY. <PF3> OR <ENTER> RETURNS TO THE ORIGINATING        */
/*   SCREEN.                                                          */
/*                                                                    */
/*   THE HIPERPOOL EFFICIENCY (HP EFF)                                */
/*                            IS CALCULATED AS FOLLOWS :-             */
/*   (HP PAGES READ / HP PAGES WRITTEN)                               */
/*   WHERE HP PAGES READ = SUM OF SYNCHRONOUS, ASYNC WITH ADM         */
/*                         AND ASYNC WITHOUT ADM PAGES READ.          */
/*         HP PAGES WRITTEN = SUM OF SYNCHRONOUS, ASYNC WITH ADM      */
/*                            AND ASYNC WITHOUT ADM PAGES             */
/*                            WRITTEN.                                */
/*         FROM THE DSNB430 AND 431 MESSAGES EXPRESSED AS A           */
/*         PERCENTAGE.                                                */
/*                                                                    */
/*   THE SCREEN(S) USE INFORMATION GLEANED FORM THE FOLLOWING         */
/*   DSNB4XXI MESSAGES :- 2, 3, 4, 5, 9, 10, 11, 12, 13, 14,          */
/*                        15, 20, 21, 30, 31 & 40.                    */
/*                                                                    */
/*   IF A BUFFERPOOL IS AMENDED, IE AN 'A' IS INSERTED, THE           */
/*   AMENDMENT SCREEN (BUFF002P) IS SHOWN. THE CURSOR IS PLACED       */
/*   ON THE FIRST AMENDABLE COLUMN. NONE, ONE, OR MORE AMENDMENTS     */
/*   ARE ALLOWED PER SCREEN. <PF3> EXITS THE PANEL WITH NO            */
/*   AMENDMENTS MADE. IF AMENDMENTS ARE MADE AND <ENTER> IS HIT       */
/*   THE SCREEN IS REDISPLAYED WITH THE ALTER COMMAND TO BE           */
/*   PROCESSED. <ENTER> WILL PROCESS THE COMMAND WITH AN              */
/*   APPROPRIATE MESSAGE, <PF3> WILL CANCEL THE COMMAND AND EXIT      */
```

```
/*   THE SCREEN.                                                    */
/*                                                                  */
/*******************************************************************/
   ADDRESS "ISPEXEC" "CONTROL ERRORS RETURN"
   ADDRESS "ISPEXEC" "TBCREATE BPLIST"||,
       " KEYS(BPOOL)"||,
       " NAMES(VPSIZE HPSIZE GETPGS SYNCIO EFF VPALLOC HPES HPOOL"||,
       " VS HS DW VW PS RGP SGP DMH IOR IOS NWE DWH VDH SPR SPP  "||,
       " SPI LPR LPP LPI DPR DPP DPI NRE NB SHR AHR NRF SHW AHW  "||,
       " NWF HPR HPW HPEFF URF UWF PRR DP O)                     "||,
       " NOWRITE REPLACE"
   DO FOREVER
      ADDRESS "ISPEXEC" "TBDISPL BPLIST PANEL(BUFFØØ1P)"
      IF RC > 4 THEN
         DO
           ADDRESS "ISPEXEC" "TBEND BPLIST"
           EXIT
         END
      ADDRESS "ISPEXEC" "VPUT (ZZSSID) PROFILE"
      IF ZTDSELS > Ø THEN
         DO
           CALL GET_BPLIST_ROW
           IF A = 'A' THEN
              CALL PROCESS_BP_A
              ELSE
              CALL PROCESS_BP_S
         END
      CALL BUILD_BPLIST;
   END
   EXIT
/*******************************************************************/
/* BUILD_BPLIST                                                    */
/*******************************************************************/
BUILD_BPLIST:
    BPCOUNT = Ø
    NEWSTACK
    DUMMY = OUTTRAP("OUTLINE.","*")
    IF SHOWACT = 'A' & INCORCUM = 'C'
       THEN
       QUEUE "-DIS BPOOL(ACTIVE) DETAIL(*)"
       ELSE
       IF SHOWACT = 'A' & INCORCUM = 'I'
          THEN
          QUEUE "-DIS BPOOL(ACTIVE) DETAIL(INTERVAL)"
          ELSE
          IF SHOWACT = 'O' & INCORCUM = 'C'
             THEN
             QUEUE "-DIS BPOOL(*) DETAIL(*)"
             ELSE
             IF SHOWACT = 'O' & INCORCUM = 'I'
                THEN
```

```
            QUEUE "-DIS BPOOL(*) DETAIL(INTERVAL)"
 IF INCORCUM = 'C' THEN
    TYPE = ' CUMULATIVE'
    ELSE
    TYPE = 'INCREMENTAL'
 QUEUE "END"
 "DSN SYSTEM("ZZSSID")"
 IF RC > Ø THEN
    DO
      SAY 'RC='||RC
      DO I = 1 TO OUTLINE.Ø
         SAY OUTLINE.I
      END
      RETURN
    END
 N = 1
 DO I = 1 TO OUTLINE.Ø
    CALL PARSE_DETAIL
    N = N + 1
 END
 N = N - 1
ADDRESS "ISPEXEC" "TBCREATE BPLIST"||,
    " KEYS(BPOOL)"||,
    " NAMES(VPSIZE HPSIZE GETPGS SYNCIO EFF VPALLOC HPES HPOOL"||,
    " VS HS DW VW PS RGP SGP DMH IOR IOS NWE DWH VDH SPR SPP  "||,
    " SPI LPR LPP LPI DPR DPP DPI NRE NB SHR AHR NRF SHW AHW  "||,
    " NWF HPR HPW HPEFF URF UWF PRR DP O)                     "||,
    " NOWRITE REPLACE"
IF RC > 4 THEN
   DO
   SAY 'TBCREATE ERROR '||RC
   EXIT
   END
DO I = 1 TO BPCOUNT
   A = ''
   BPOOL     = TBPOOL.I
   VPSIZE    = TVPSIZE.I
   HPSIZE    = THPSIZE.I
   GETPGS    = TGETPGS.I
   SYNCIO    = TSYNCIO.I
   ASYNC     = TASYNC.I
   VPALLOC   = TVPALLOC.I
   HPES      = THPES.I
   HPOOL     = THPNAME.I
   VS        = TVPSEQ.I
   HS        = THPSEQ.I
   DW        = TDEFWR.I
   VW        = TVDEFWR.I
   PS        = TPASEQ.I
   RGP       = TRGETPGS.I
   SGP       = TSGETPGS.I
```

```
DMH       = TDMTHIT.I
IOR       = TSYNCIOR.I
IOS       = TSYNCIOS.I
NWE       = TNOWENG.I
DWH       = TDWTHIT.I
VDH       = TVDWTHIT.I
SPR       = TSPREQ.I
SPP       = TSPPR.I
SPI       = TSPPIO.I
LPR       = TLPREQ.I
LPP       = TLPPR.I
LPI       = TLPPIO.I
DPR       = TDPREQ.I
DPP       = TDPPR.I
DPI       = TDPPIO.I
NRE       = TNOREADENG.I
NB        = TNOBUF.I
SHR       = TSHPREADS.I
AHR       = TASHPREADS.I
NRF       = TNURFAIL.I
SHW       = TSHPWRITES.I
AHW       = TASHPWRITES.I
NWF       = TNUWFAIL.I
HPR       = THPREADS.I
HPW       = THPWRITES.I
URF       = TURFAIL.I
UWF       = TUWFAIL.I
PRR       = TPRLREQ.I
DP        = TDEGPAR.I
O         = TCO.I
IF SYNCIO = Ø THEN
   EFF    = '*****'
   ELSE
   EFF    = ((GETPGS - SYNCIO - ASYNC)/GETPGS)*1ØØ
IF EFF    > 99999 THEN
   EFF    = '*****'
SUMHW = SHW + AHW + HPW
IF SUMHW > Ø
   THEN
        HPEFF  = ((SHR + AHR + HPR)/(SHW + AHW + HPW))*1ØØ
   ELSE
        HPEFF = Ø
IF VPSIZE > Ø
   THEN
   DO
   ADDRESS "ISPEXEC" "TBADD BPLIST"
   IF RC > Ø THEN
      DO
      SAY 'TBADD ERROR '||RC
      ADDRESS "ISPEXEC" "TBEND BPLIST"
      EXIT
```

```
            END
          END
   END
   ADDRESS "ISPEXEC" "TBSORT BPLIST "||,
                     "FIELDS(BPOOL,C,A)"
   ADDRESS "ISPEXEC" "TBTOP BPLIST"
   IF RC > 4 THEN
       DO
       SAY 'TBTOP ERROR '||RC
       ADDRESS "ISPEXEC" "TBEND BPLIST"
       EXIT
       END
RETURN;
/*********************************************************************/
/* PARSE_DETAIL                                                      */
/*********************************************************************/
PARSE_DETAIL:
  IF SUBSTR(OUTLINE.I,Ø1,Ø8) = 'DSNB4Ø1I' THEN
     DO
       PARSE VAR OUTLINE.I S1 S2 S3 S4 S5 S6 S7 S8 S9 SA SB
       STRLEN              = LENGTH(S5)
       BPCOUNT             = BPCOUNT + 1
       TBPOOL.BPCOUNT      = SUBSTR(S5,Ø1,STRLEN-1)
       TVPSIZE.BPCOUNT     = Ø
       TVPALLOC.BPCOUNT    = Ø
       TVPDEL.BPCOUNT      = Ø
       TVPINUSE.BPCOUNT    = Ø
       THPSIZE.BPCOUNT     = Ø
       THPES.BPCOUNT       = Ø
       THPDEL.BPCOUNT      = Ø
       THPALLOC.BPCOUNT    = Ø
       THPNAME.BPCOUNT     = '        '
       TGETPGS.BPCOUNT     = Ø
       TSYNCIO.BPCOUNT     = Ø
       TDMTHIT.BPCOUNT     = Ø
       TVPSEQ.BPCOUNT      = Ø
       THPSEQ.BPCOUNT      = Ø
       TDEFWR.BPCOUNT      = Ø
       TVDEFWR.BPCOUNT     = Ø
       TPASEQ.BPCOUNT      = Ø
       TDPREQ.BPCOUNT      = Ø
       TDPPIO.BPCOUNT      = Ø
       TDPPR.BPCOUNT       = Ø
       TLPREQ.BPCOUNT      = Ø
       TLPPIO.BPCOUNT      = Ø
       TLPPR.BPCOUNT       = Ø
       TSPREQ.BPCOUNT      = Ø
       TSPPIO.BPCOUNT      = Ø
       TSPPR.BPCOUNT       = Ø
       TBPHITR.BPCOUNT     = Ø
       TNOBUF.BPCOUNT      = Ø
```

```
      TNOREADENG.BPCOUNT  = Ø
      TRGETPGS.BPCOUNT    = Ø
      TSGETPGS.BPCOUNT    = Ø
      TCO.BPCOUNT         = ' '
      TSYNCIOR.BPCOUNT    = Ø
      TSYNCIOS.BPCOUNT    = Ø
      TNOWENG.BPCOUNT     = Ø
      TDWTHIT.BPCOUNT     = Ø
      TVDWTHIT.BPCOUNT    = Ø
      TSHPREADS.BPCOUNT   = Ø
      TASHPREADS.BPCOUNT  = Ø
      TNURFAIL.BPCOUNT    = Ø
      TSHPWRITES.BPCOUNT  = Ø
      TASHPWRITES.BPCOUNT = Ø
      TNUWFAIL.BPCOUNT    = Ø
      THPREADS.BPCOUNT    = Ø
      THPWRITES.BPCOUNT   = Ø
      TURFAIL.BPCOUNT     = Ø
      TUWFAIL.BPCOUNT     = Ø
      TPRLREQ.BPCOUNT     = Ø
      TDEGPAR.BPCOUNT     = Ø
    END
  IF SUBSTR(OUTLINE.I,Ø1,Ø8) = 'DSNB4Ø2I' THEN
    DO
      J = I
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8
      TVPSIZE.BPCOUNT     = S7
      J = J + 1
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8 S9
      TVPALLOC.BPCOUNT    = S3
      TVPDEL.BPCOUNT      = S8
      J = J + 1
      PARSE VAR OUTLINE.J S1 S2 S3 S4
      TVPINUSE.BPCOUNT    = S3
    END
  IF SUBSTR(OUTLINE.I,Ø1,Ø8) = 'DSNB4Ø3I' THEN
    DO
      J = I
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8 S9 SA SB
      THPSIZE.BPCOUNT     = S6
      TCO.BPCOUNT         = SA
      J = J + 1
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8 S9
      THPALLOC.BPCOUNT    = S3
      THPDEL.BPCOUNT      = S8
      J = J + 1
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6
      THPES.BPCOUNT       = S5
    END
  IF SUBSTR(OUTLINE.I,Ø1,Ø8) = 'DSNB4Ø4I' THEN
    DO
```

```
      J = I + 1
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8 S9 SA
      TVPSEQ.BPCOUNT      = S4
      THPSEQ.BPCOUNT      = S8
      J = J + 1
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8 S9 SA
      TDEFWR.BPCOUNT      = S4
      TVDEFWR.BPCOUNT     = S9
      J = J + 1
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8 S9 SA
      TPASEQ.BPCOUNT      = S4
    END
  IF SUBSTR(OUTLINE.I,Ø1,Ø8) = 'DSNB4Ø5I' THEN
    DO
      J = I
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7
      THPNAME.BPCOUNT     = S6
    END
  IF SUBSTR(OUTLINE.I,Ø1,Ø8) = 'DSNB4Ø9I' THEN
    DO
      PARSE VAR OUTLINE.I S1 S2 S3 S4 S5 S6
      STATTIME            = S6
    END
  IF SUBSTR(OUTLINE.I,Ø1,Ø8) = 'DSNB41ØI' THEN
    DO
      PARSE VAR OUTLINE.I S1 S2 S3 S4 S5 S6
      STATTIME            = S6
    END
  IF SUBSTR(OUTLINE.I,Ø1,Ø8) = 'DSNB411I' THEN
    DO
      J = I
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8 S9 SA SB SC SD
      TGETPGS.BPCOUNT     = S6
      TRGETPGS.BPCOUNT    = S6
      J = J + 1
      PARSE VAR OUTLINE.J S1
      TSYNCIO.BPCOUNT     = S1
      TSYNCIOR.BPCOUNT    = S1
      J = J + 1
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8 S9 SA SB
      TGETPGS.BPCOUNT     = TGETPGS.BPCOUNT + S4
      TSGETPGS.BPCOUNT    = S4
      TSYNCIO.BPCOUNT     = TSYNCIO.BPCOUNT + SA
      TSYNCIOS.BPCOUNT    = SA
      J = J + 1
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5
      TDMTHIT.BPCOUNT     = S4
    END
  IF SUBSTR(OUTLINE.I,Ø1,Ø8) = 'DSNB412I' THEN
    DO
      J = I + 1
```

```
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8
      TSPREQ.BPCOUNT      = S3
      TSPPIO.BPCOUNT      = S7
      J = J + 1
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5
      TSPPR.BPCOUNT       = S4
      TASYNC.BPCOUNT      = S4
    END
  IF SUBSTR(OUTLINE.I,Ø1,Ø8) = 'DSNB413I' THEN
    DO
      J = I + 1
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8
      TLPREQ.BPCOUNT      = S3
      TLPPIO.BPCOUNT      = S7
      J = J + 1
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5
      TLPPR.BPCOUNT       = S4
      TASYNC.BPCOUNT      = TASYNC.BPCOUNT + S4
    END
  IF SUBSTR(OUTLINE.I,Ø1,Ø8) = 'DSNB414I' THEN
    DO
      J = I + 1
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8
      TDPREQ.BPCOUNT      = S3
      TDPPIO.BPCOUNT      = S7
      J = J + 1
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5
      TDPPR.BPCOUNT       = S4
      TASYNC.BPCOUNT      = TASYNC.BPCOUNT + S4
    END
  IF SUBSTR(OUTLINE.I,Ø1,Ø8) = 'DSNB415I' THEN
    DO
      J = I + 1
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8 S9 SA
      TNOBUF.BPCOUNT      = S4
      TNOREADENG.BPCOUNT  = S9
    END
  IF SUBSTR(OUTLINE.I,Ø1,Ø8) = 'DSNB42ØI' THEN
    DO
      J = I
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8 S9 SA
      TSYSP.BPCOUNT       = S7
      J = J + 1
      PARSE VAR OUTLINE.J S1
      TSYSPW.BPCOUNT      = S1
      J = J + 1
      PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8 S9 SA
      TASYNCWIO.BPCOUNT   = S5
      TSYNCWIO.BPCOUNT    = SA
    END
  IF SUBSTR(OUTLINE.I,Ø1,Ø8) = 'DSNB421I' THEN
```

```
      DO
        J = I
        PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8 S9 SA
        TDWTHIT.BPCOUNT     = S6
        J = J + 1
        PARSE VAR OUTLINE.J S1
        TVDWTHIT.BPCOUNT    = S6
        J = J + 1
        PARSE VAR OUTLINE.J S1 S2 S3 S4 S5
        TNOWENG.BPCOUNT     = S5
      END
    IF SUBSTR(OUTLINE.I,Ø1,Ø8) = 'DSNB43ØI' THEN
      DO
        J = I + 2
        PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8 S9 SA
        TSHPREADS.BPCOUNT   = S5
        TSHPWRITES.BPCOUNT  = SA
        J = J + 1
        PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8 S9 SA
        TASHPREADS.BPCOUNT  = S5
        TASHPWRITES.BPCOUNT = SA
        J = J + 1
        PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8
        TNURFAIL.BPCOUNT    = S4
        TNUWFAIL.BPCOUNT    = S8
      END
    IF SUBSTR(OUTLINE.I,Ø1,Ø8) = 'DSNB431I' THEN
      DO
        J = I + 2
        PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8
        THPREADS.BPCOUNT    = S4
        THPWRITES.BPCOUNT   = S8
        J = J + 1
        PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7 S8
        TURFAIL.BPCOUNT     = S4
        TUWFAIL.BPCOUNT     = S8
      END
    IF SUBSTR(OUTLINE.I,Ø1,Ø8) = 'DSNB44ØI' THEN
      DO
        J = I + 1
        PARSE VAR OUTLINE.J S1 S2 S3 S4 S5 S6 S7
        TPRLREQ.BPCOUNT     = S4
        TDEGPAR.BPCOUNT     = S7
      END
RETURN;
/*********************************************************************/
/* GET_BPLIST_ROW                                                    */
/*********************************************************************/
GET_BPLIST_ROW:
  ADDRESS "ISPEXEC" "TBGET BPLIST"
RETURN;
```

```
/*********************************************************************/
/* PROCESS_BP_A                                                      */
/*********************************************************************/
PROCESS_BP_A:
  ALTERØ1   = ''
  ALTERØ2   = ''
  ALTERØ3   = ''
  NVPSIZE   = ''
  NHPSIZE   = ''
  NVPSEQ    = ''
  NHPSEQ    = ''
  NDEFWR    = ''
  NVDEFWR   = ''
  NPASEQ    = ''
  NCASTOUT  = ''
  DO FOREVER
   ADDRESS "ISPEXEC" "DISPLAY PANEL (BUFFØØ2P)"
   IF RC > 4 THEN RETURN
   ADDRESS "ISPEXEC" "CONTROL DISPLAY LINE START(45)"
   UPDSW  = 'N'
   IF NVPSIZE  > '' THEN UPDSW = 'Y'
   IF NHPSIZE  > '' THEN UPDSW = 'Y'
   IF NVPSEQ   > '' THEN UPDSW = 'Y'
   IF NHPSEQ   > '' THEN UPDSW = 'Y'
   IF NDEFWR   > '' THEN UPDSW = 'Y'
   IF NVDEFWR  > '' THEN UPDSW = 'Y'
   IF NPASEQ   > '' THEN UPDSW = 'Y'
   IF NCASTOUT > '' THEN UPDSW = 'Y'
   ALTSTRING = '-ALTER BPOOL('||BPOOL||')'
   ALTSTRING = ALTSTRING||' '
   IF NVPSIZE  > '' THEN
      DO
      ALTSTRING = ALTSTRING||'VPSIZE('||STRIP(NVPSIZE,,' ')||')'
      ALTSTRING = ALTSTRING||' '
      END
   IF NHPSIZE  > '' THEN
      DO
      ALTSTRING = ALTSTRING||'HPSIZE('||STRIP(NHPSIZE,,' ')||')'
      ALTSTRING = ALTSTRING||' '
      END
   IF NVPSEQ   > '' THEN
      DO
      ALTSTRING = ALTSTRING||'VPSEQT('||STRIP(NVPSEQ,,' ')||')'
      ALTSTRING = ALTSTRING||' '
      END
   IF NHPSEQ   > '' THEN
      DO
      ALTSTRING = ALTSTRING||'HPSEQT('||STRIP(NHPSEQ,,' ')||')'
      ALTSTRING = ALTSTRING||' '
      END
   IF NDEFWR   > '' THEN
```

```
   DO
   ALTSTRING = ALTSTRING||'DWQT('||STRIP(NDEFWR,,' ')||')'
   ALTSTRING = ALTSTRING||' '
   END
IF NVDEFWR  > '' THEN
   DO
   ALTSTRING = ALTSTRING||'VDWQT('||STRIP(NVDEFWR,,' ')||')'
   ALTSTRING = ALTSTRING||' '
   END
IF NPASEQ   > '' THEN
   DO
   ALTSTRING = ALTSTRING||'VPPSEQT('||STRIP(NPASEQ,,' ')||')'
   ALTSTRING = ALTSTRING||' '
   END
IF NCASTOUT > '' THEN
   DO
   ALTSTRING = ALTSTRING||'CASTOUT('||STRIP(NCASTOUT,,' ')||')'
   ALTSTRING = ALTSTRING||' '
   END
IF UPDSW = 'Y' THEN
   DO
      ALTERØ1 = 'THE FOLLOWING ALTER WILL BE PERFORMED :-'
      ALTERØ2 = SUBSTR(ALTSTRING,Ø1,7Ø)
      ALTERØ3 = SUBSTR(ALTSTRING,71,7Ø)
      ADDRESS "ISPEXEC" "DISPLAY PANEL (BUFFØØ2P)"
      IF RC > 4 THEN RETURN
      NEWSTACK
      DUMMY = OUTTRAP("ALTOUT.","*")
      QUEUE ALTSTRING
      QUEUE "END"
      "DSN SYSTEM("ZZSSID")"
      IF RC = Ø THEN
         DO
            ALTERØ1 = 'ALTER PERFORMED SUCCESSFULLY'
            ALTERØ2 = ''
            ALTERØ3 = ''
            ADDRESS "ISPEXEC" "DISPLAY PANEL (BUFFØØ2P)"
            IF RC > 4 THEN
               DO
                  ALTERØ1 = ''
                  ALTERØ2 = ''
                  ALTERØ3 = ''
                  RETURN
               END
         END
      ELSE
         DO
 IF RC > Ø THEN
    DO
      SAY 'RC='||RC
      DO I = 1 TO ALTOUT.Ø
```

```
            SAY ALTOUT.I
          END
          RETURN
        END
              ALTERØ1 = 'ALTER FAILURE'
              ALTERØ2 = ''
              ALTERØ3 = ''
              ADDRESS "ISPEXEC" "DISPLAY PANEL (BUFFØØ2P)"
              IF RC > 4 THEN
                 DO
                    ALTERØ1 = ''
                    ALTERØ2 = ''
                    ALTERØ3 = ''
                    RETURN
                 END
            END
  END
ALTERØ1 = ''
ALTERØ2 = ''
ALTERØ3 = ''
RETURN;
/*********************************************************************/
/* GET_ROW                                                           */
/*********************************************************************/
PROCESS_BP_S:
   ADDRESS "ISPEXEC" "DISPLAY PANEL (BUFFØØ3P)"
   IF RC > 4 THEN RETURN
RETURN;
```

BUFFP001P

```
)ATTR
/*********************************************************************/
/* BPOOLP - DB2 BUFFERPOOL MAINTENANCE UTILITY                       */
/*********************************************************************/
   _ TYPE(INPUT) INTENS(HIGH)  CAPS(ON)           JUST(LEFT )
   + TYPE(TEXT)   INTENS(LOW)   COLOR(BLUE)        SKIP(ON)
   ? TYPE(TEXT)   INTENS(LOW)   COLOR(TURQUOISE)   SKIP(ON)
   % TYPE(TEXT)   INTENS(HIGH)  COLOR(YELLOW)      SKIP(ON)
   ¬ TYPE(OUTPUT) INTENS(HIGH)  COLOR(YELLOW)      CAPS(ON) JUST(RIGHT)
   # TYPE(OUTPUT) INTENS(HIGH)  COLOR(GREEN)       CAPS(ON) JUST(RIGHT)
   ! TYPE(OUTPUT) INTENS(HIGH)  COLOR(GREEN)       CAPS(ON) JUST(LEFT)
)BODY CMD(C)
%----------------------+DB2 BUFFERPOOL MAINTENANCE%-------------------
+OPTION ===>_C                                              +SCR->_AMT +
+                                                        +DB2 SSID: _Z   +
%ACTIVE OR ALL DEFINED    :_Z? (ENTER A FOR ACTIVE OR O FOR ALL)       +
%INCREMENTAL OR CUMULATIVE:_Z? (ENTER I FOR INCREMENTAL OR C FOR
CUMULATIVE) +
+                                                                      +
```

```
+  ¬TYPE       %BUFFERPOOL STATISTICS SINCE:!STATTIME                  +
+                                                                      +
%   BUFFER     <----SIZE---->   NUMBER       VP   <------DEFAULTS----->+
%    POOL      VIRTUAL  HIPER     OF         SYNC   READ   VP  HP DEF
VDEF PA C
%C   NAME       POOL     POOL  GETPAGES     READS    EFF  SEQ SEQ WRT
WRT SEQ O
%---------------------------------------------------------------------+
)MODEL
_A+ !BPOOL+#VPSIZE   #HPSIZE #GETPGS   #SYNCIO    #EFF  #VS #HS #DW #VW
#PS +#O+
)INIT
 .ZVARS = '(ZZSSID, SHOWACT, INCORCUM)'
 .CURSOR = SHOWACT
)PROC
  VER  (&C LIST,END,' ')
  VER  (&ZZSSID NONBLANK LIST,DB2P,DB2U,DB2D,DB2T,DGDD,DGDC,DGDP)
  VER  (&SHOWACT NONBLANK LIST,A,O)
  VER  (&INCORCUM NONBLANK LIST,I,C)
)END
```

BUFFP002P

```
)ATTR
/*********************************************************************/
/* BPOOLP2 - DB2 BUFFERPOOL MAINTENANCE UTILITY, UPDATE PANEL        */
/*********************************************************************/
   + TYPE(TEXT)   INTENS(LOW)   COLOR(BLUE)        SKIP(ON)
   ¬ TYPE(OUTPUT) INTENS(HIGH)  COLOR(GREEN)       SKIP(ON) JUST(LEFT)
   % TYPE(TEXT)   INTENS(HIGH)  COLOR(WHITE)       SKIP(ON)
   ? TYPE(TEXT)   INTENS(HIGH)  COLOR(RED)         SKIP(ON)
   # TYPE(OUTPUT) INTENS(HIGH)  COLOR(TURQUOISE)   CAPS(ON) JUST(RIGHT)
   / TYPE(OUTPUT) INTENS(HIGH)  COLOR(RED)         CAPS(ON) JUST(LEFT)
   ! TYPE(TEXT)   INTENS(HIGH)  COLOR(TURQUOISE)   SKIP(ON)
   @ TYPE(TEXT)   INTENS(HIGH)  COLOR(YELLOW)      SKIP(ON)
)BODY CMD(C2)
%----------------------+DB2 BUFFERPOOL MAINTENANCE%--------------------
%OPTION ===>_C2                                        +DB2 SSID: _Z   +
+
% BUFFERPOOL: ¬BPOOL   +  !CURRENT VALUE       NEW VALUE
%                          =============       =========
@(SIZES)
+      VIRTUAL POOL SIZE     ¬VPSIZE+          _NVPSIZE+
+        HIPER POOL SIZE     ¬HPSIZE+          _NHPSIZE+
@(THRESHOLDS)
+          VP SEQUENTIAL     ¬VS               _Z +
+          HP SEQUENTIAL     ¬HS               _Z +
+         DEFERRED WRITE     ¬DW               _Z +
+  VERTICAL DEFERRED WRT     ¬VW               _Z +
```

```
+    PARALLEL SEQUENTIAL     ¬PS               _Z +
@(ATTRIBUTES)
+                CASTOUT     ¬O                _Z  +
+
? INSTRUCTIONS :- TO ALTER VALUE(S) : INSERT NEW VALUE(S), PRESS ENTER
?                 TO EXIT SCREEN    : PRESS <PF3>
+
 /ALTERØ1
+
+
 ¬ALTERØ2
+
 ¬ALTERØ3
+
+
)INIT
 .ZVARS = '(ZZSSID, NVPSEQ, NHPSEQ NDEFWR, NVDEFWR, NPASEQ, NCASTOUT)'
 .CURSOR = NVPSIZE
)PROC
  VER  (&C2 LIST,END,ALTER,' ')
  VER  (&NVPSIZE NUM)
  VER  (&NHPSIZE NUM)
  VER  (&NVPSEQ  RANGE,Ø,1ØØ)
  VER  (&NHPSEQ  RANGE,Ø,1ØØ)
  VER  (&NDEFWR  RANGE,Ø,1ØØ)
  VER  (&NVDEFWR  RANGE,Ø,1ØØ)
  VER  (&NPASEQ  RANGE,Ø,1ØØ)
  VER  (&NCASTOUT LIST,YES,NO)
)END
```

BUFF003P

```
)ATTR
/*********************************************************************/
/* BPOOLP2 - DB2 BUFFERPOOL MAINTENANCE UTILITY, UPDATE PANEL        */
/*********************************************************************/
   + TYPE(TEXT)   INTENS(LOW)   COLOR(BLUE)        SKIP(ON)
   ¬ TYPE(OUTPUT) INTENS(HIGH)  COLOR(GREEN)       SKIP(ON) JUST(LEFT)
   % TYPE(TEXT)   INTENS(HIGH)  COLOR(WHITE)       SKIP(ON)
   # TYPE(OUTPUT) INTENS(HIGH)  COLOR(TURQUOISE)   CAPS(ON) JUST(RIGHT)
   ! TYPE(TEXT)   INTENS(HIGH)  COLOR(TURQUOISE)   SKIP(ON)
   @ TYPE(TEXT)   INTENS(HIGH)  COLOR(YELLOW)      SKIP(ON)
)BODY CMD(C2A)
%--------------------+DB2 BUFFERPOOL MAINTENANCE%---------------------
%OPTION ===>_C2A                                      +DB2 SSID: #Z    +
+                                                                      +
+BUFFERPOOL:  ¬BPOOL   + HIPERPOOL: ¬HPOOL       +  CASTOUT:    ¬O     +
@(SIZES)
+POOL SIZE : VIRTUAL      - ¬VPSIZE  + HIPER        - ¬HPSIZE +
+            ALLOCATED    - ¬VPALLOC + BACKED BY ES - ¬HPES     +
```

```
@(VPSTATS)
+GETPAGE         :  RANDOM - ¬RGP     + SEQN     - ¬SGP     + DMTH HIT -
¬DMH    +
+SYNC READ (I/O):  R      - ¬IOR     + S        - ¬IOS     +
+NO WRITE ENGINE           - ¬NWE     + DWT HIT - ¬DWH     + VDWT HIT -
¬VDH    +
@(PREFETCH)
+SEQN: REQUESTS -  ¬SPR    + PAGES READ    -  ¬SPP      + I/O       -
¬SPI    +
+LIST: REQUESTS -  ¬LPR    + PAGES READ    -  ¬LPP      + I/O       -
¬LPI    +
+DYN : REQUESTS -  ¬DPR    + PAGES READ    -  ¬DPP      + I/O       -
¬DPI    +
+PREFETCH DISABLED NO :      READ ENGINE   -  ¬NRE      + BUFFERS   -
¬NB     +
@(HPSTATS - NOT USING ASYNCHRONOUS DATA MOVER FACILITY)
+READS  : SYNC  -  ¬SHR    + ASYNC         -  ¬AHR      + FAILURES  -
¬NRF    +
+WRITES : SYNC  -  ¬SHW    + ASYNC         -  ¬AHW      + FAILURES  -
¬NWF    +
@(HPSTATS - USING ASYNCHRONOUS DATA MOVER FACILITY)
+READS          -  ¬HPR    + WRITES        -  ¬HPW      +
+READ FAILURES  -  ¬URF    + WRITE FAILURES-  ¬UWF      + HP EFF    -
¬HPEFF  +
@(PARALLEL ACTIVITY)
+REQUESTS       -  ¬PRR    + DEGRADED      -  ¬DP       +
)INIT
 .ZVARS = '(ZZSSID)'
)PROC
  VER  (&C2 LIST,END,ALTER,' ')
)END
```

Ian McDonald
DB2 DBA (UK)

A simple SQL query for tuning indexes

An index can provide efficient access to data. In fact, that is the only purpose of non-unique indexes. Unique indexes have the additional function of ensuring that key values are unique.

A table can have more than one index, and an index key can use one or more columns. An index key is a column or an ordered collection

of columns on which an index is defined. The usefulness of an index depends on its key. Columns that you use frequently in performing selection, join, grouping, and ordering operations are good candidates for use as keys.

But before you create an index, consider carefully its cost:

- Indexes require storage space.
- Each index requires an index space and a dataset.
- Locking problems with more indexes on the same table.
- An index must be changed to reflect every insert or delete operation on the base table.
- Indexes can be built automatically when loading a table, but this takes time.

My example is from the QMF environment:

PROC: PEXP

```
RUN QUERY userid.QEXP (FORM userid.FEXP
```

QUERY: QEXP

```
SELECT IXNAME, COLNAME, COLSEQ, COUNT(*)
FROM SYSIBM.SYSKEYS,user.PLAN_TABLE
WHERE IXCREATOR = userid  <--- insert your userid
  AND IXNAME LIKE &INDEX
  AND ACCESSTYPE='I'
  AND ACCESSNAME=IXNAME
  AND COLSEQ=MATCHCOLS
GROUP BY IXNAME,COLNAME,COLSEQ
UNION
SELECT IXNAME, COLNAME, COLSEQ, 0
FROM SYSIBM.SYSKEYS K
WHERE IXCREATOR = userid  <--- insert your userid
AND IXNAME LIKE &INDEX
AND NOT EXISTS (SELECT * FROM user.PLAN_TABLE
                WHERE ACCESSTYPE='I'
                  AND ACCESSNAME=K.IXNAME
                  AND MATCHCOLS=K.COLSEQ)
ORDER BY 1,3
```

The MATCHCOLS column in PLAN_TABLE shows how many of the index columns are matched by predicates.

Statistics from DB2 catalog and PLAN_TABLE help determine the most economical index.

FORM: FEXP

```
H QMF Ø8 F Ø4 E V W E R Ø1 Ø3 97/Ø7/Ø4 1Ø:53
T 1111Ø ØØ4 Ø11 1112 ØØ7 1113 Ø4Ø 1114 ØØ7 1115 ØØ6 1116 ØØ5 1117 ØØ5
1118 ØØ3 1119 ØØ8 112Ø ØØ8 1122 ØØ6 1121 Ø5Ø
R CHAR    INDEX                                      BREAK1  2      1Ø
C     1   DEFAULT  DEFAULT  NO
R CHAR    COLUMN                                             2      1Ø
C     2   DEFAULT  DEFAULT  NO
R NUMERIC MATCHCOLS                                          3      9
L     3   DEFAULT  DEFAULT  NO
R NUMERIC SUM                                        SUM     3      6
L     4   DEFAULT  DEFAULT  NO
V 12Ø1 ØØ1 Ø
V 12Ø2 ØØ1 1
T 121Ø ØØ1 ØØ3 1212 ØØ4 1213 ØØ6 1214 Ø55
R 1    3      DISPLAY RELATION BETWEEN INDEKSES/MATCHCOLS
V 13Ø1 ØØ1 2
V 13Ø2 ØØ1 Ø
T 131Ø ØØ1 ØØ3 1312 ØØ4 1313 ØØ6 1314 Ø55
R 1    CENTER
V 14Ø1 ØØ2 NO
V 14Ø2 ØØ1 1
V 14Ø3 ØØ1 Ø
T 141Ø ØØ3 ØØ3 1412 ØØ4 1413 ØØ6 1414 Ø55
R 1    RIGHT  SUM TOTAL:
R 2    RIGHT
R 4    RIGHT  QMF PROCEDURE "PEXP"  1995,"ZB"
V 15Ø1 ØØ1 1
V 15Ø2 ØØ3 YES
V 15Ø3 ØØ3 YES
V 15Ø4 ØØ3 YES
V 15Ø5 ØØ3 YES
V 15Ø6 ØØ3 YES
V 15Ø7 ØØ3 YES
V 15Ø8 ØØ3 YES
V 15Ø9 ØØ3 YES
V 151Ø ØØ3 YES
V 1511 ØØ4 NONE
V 1512 ØØ2 NO
V 1513 ØØ7 DEFAULT
V 1514 ØØ2 NO
```

```
V 1515 ØØ4 NONE
V 279Ø ØØ1 1
V 2791 ØØ3 YES
V 28Ø5 ØØ3 YES
T 281Ø ØØ1 ØØ3 2812 ØØ4 2813 ØØ6 2814 Ø55
R 1    LEFT
V 29Ø1 ØØ2 NO
V 29Ø2 ØØ1 1
V 29Ø4 ØØ1 Ø
V 29Ø6 ØØ2 NO
V 29Ø7 ØØ2 NO
T 291Ø ØØ1 ØØ3 2912 ØØ4 2913 ØØ6 2914 Ø55
R 1    LEFT
V 3Ø8Ø ØØ1 1
V 31Ø1 ØØ2 NO
V 31Ø2 ØØ2 NO
V 31Ø3 ØØ1 Ø
V 31Ø4 ØØ1 Ø
T 311Ø ØØ1 ØØ3 3112 ØØ4 3113 ØØ6 3114 Ø55
R 1    LEFT
V 32Ø1 ØØ2 NO
V 32Ø2 ØØ1 1
V 32Ø3 ØØ1 Ø
V 32Ø4 ØØ1 Ø
T 321Ø ØØ1 ØØ3 3212 ØØ4 3213 ØØ6 3214 Ø55
R 1    RIGHT  SUM  &1:
V 3Ø8Ø ØØ1 2
V 31Ø1 ØØ2 NO
V 31Ø2 ØØ2 NO
V 31Ø3 ØØ1 Ø
V 31Ø4 ØØ1 Ø
T 311Ø ØØ1 ØØ3 3112 ØØ4 3113 ØØ6 3114 Ø55
R 1    LEFT
V 32Ø1 ØØ2 NO
V 32Ø2 ØØ1 1
V 32Ø3 ØØ1 Ø
V 32Ø4 ØØ1 1
T 321Ø ØØ1 ØØ3 3212 ØØ4 3213 ØØ6 3214 Ø55
R 1    RIGHT
E
```

Run the QMF PEXP procedure frequently after bind plans and packages. The report, displayed below, is used to determine whether matching index scans are found in the corresponding columns of the SYSIBM.SYSKEYS catalog table and PLAN_TABLE. Monitoring these statistics can help you determine if your indexes are working effectively.

REPORT

```
DISPLAY RELATION BETWEEN INDEXES/MATCHCOLS

INDEX       COLUMN       MATCHCOLS      SUM
----------  ----------   ---------   ------
 X1         COL1                 1       34    row  1
            COL2                 2      332    row  2
                                       ------
                          SUM X1:      366
 X2         COL1                 1       12
            COL2                 2       11
            COL3                 3       11
                                       ------
                          SUM X2:       34
 X3         COL1                 1        Ø
            COL3                 2        8    row  8
            COL4                 3        Ø    row  9
                                       ------
                          SUM X3:        8
                                       =======
                        SUM TOTAL:      4Ø8
```

Explanation

The index X1 has a composite key, containing columns COL1 and COL2. The value 34 (row 1) shows how many times (in plans and packages) MATCHCOLS for index X1 is 1. In row 2, MATCHCOLS=2 occurs 332 times and this means that the degree of filtering is high and the matching index scan is efficient.

The index X3 has a composite key, containing columns COL1, COL3, and COL4. Row 8 shows that MATCHCOLS=2 occurs eight times, but in row 9 you can see that MATCHCOLS=3 is never used – and this is bad. In this case, index X3 is a candidate for recreation with a composite key, using only columns COL1 and COL3.

Zver and Lorencin
DB2 Team
Informatika Maribor (Slovenia)

SMS/DB2 DBA tip

When Rudyard Kipling (1865-1936) wrote in *The Ballad of East and West* "Oh, East is East, and West is West, and never the twain shall meet", he certainly was not referring to the relationship between the DASD Storage Administrator (SA) and the Database Administrator (DBA).

Kipling's remark, however, could describe perfectly the atmosphere of 'mistrust' and 'apprehension' that exists between these two groups of computer professionals over the subject of SMS totally managing DB2 datasets.

The DBA insists that DB2 data and indexes should be placed on separate DASD volumes to avoid DASD arm movement contention. This will achieve optimum system performance objectives, enhance availability, and meet users' service level agreements.

The DBA is also concerned that certain DB2 performance enhancements, such as I/O parallelism on DB2 partitions, require by definition that each partition of a DB2 table be located on a separate volume, a requirement that SMS cannot meet.

On the other hand, the storage administrator is under increasing pressure to manage the ever growing DB2 databases. He/she wants to offload the inefficient manual tasks of managing the DASD space to SMS and argues, not without merit, that the improvement in cacheing technology at the DASD controller level and the reliability of RAID devices render the physical location of data to be of limited importance.

This article is not about the pros and cons of SMS managing DB2 datasets, so I will not enter into the debate.

I believe in a middle-of-the-road solution. I believe that the SA and the DBA should meet half way, communicate, and interact extensively with each other and learn from each other.

For example the proper usage of Automatic Class Selection (ACS) routines and filtering datasets by names or characteristics (prepared by the SA in consultation with the DB2 DBA) can benefit both

professionals and can improve data and storage management functions for their organization.

In this article, however, I shall show my DBA colleagues how SMS can help the DBA in quickly resolving an annoying nightmare that happens frequently in our DBA work.

If Murphy's Law has something to do with it, this nightmare usually happens at 2:00 am when the DBA's pager wakes him up from a deep sleep and a production support person at the other end of the phone line sadistically and in a clearly unrepentant voice relays message DSNP0071 with reason code 00D70014 that DB2 dataset X has reached 119 extents.

It is assumed that the reader is aware of the impact of DB2 dataset reaching 119 extents. It is also assumed that the reader knows why sometimes DB2 dataset might reach the maximum VSAM limit of 123 extents.

Before I show you how SMS can help the DBA to resolve this problem very quickly, let me briefly explain, at a very high level, the world of the storage administrator, and relate it to DB2 whenever I can.

The world of the storage administrator broadly consists of one or more of the following four Data Facility Storage Management Subsystem (DFSMS/MVS) products licensed from IBM. The four products are lumped together under the general name of DFSMS, but still can be licensed individually. Note that only the functionality that relates to DB2 is discussed here.

DFSMSdfp

DFSMSdfp contains, among other things, the access methods associated with dataset structures.

The Media Manager component of this product handles I/O requests from the DB2 Buffer Manager.

The VSAM component of this product handles DB2 Bootstrap Dataset (BSDS) and DB2 Active logs.

The DFSORT component of this product is invoked by various DB2 utilities such as Reorg, Load, Check, etc.

DFSMSdss

DFSMSdss provides the dataset services of the facility. Its purpose is to back-up and recover datasets and DASD volumes – at high speed.

DB2 datasets, like any other datasets, can be copied, dumped, or restored using this product. However DB2 did not become aware of these activities until Version 4, where the back-ups taken by DFSMSdss can be registered in SYSIBM.SYSCOPY and can be used by DB2 Recover Utility.

An interesting feature of this product that is related to the previous paragraph is the Concurrent Copy of DB2. This is not going to be discussed here because it is not relevant to the subject of this article.

DFSMShsm

DFSMShsm is the hierarchical storage manager of the facility. Its purpose is to manage the storage hierarchy by migrating or recalling datasets to various levels of the hierarchy, based on the activity level of the data:

- Active data or frequently accessed data is said to be at Level 0.
- Low activity data is user data that is eligible to be migrated or has been migrated. This is said to be at Level 1.
- Totally inactive data is said to be at level 2, which is usually a tape.

Each level of data is associated with a particular class of storage device.

This component can also invoke the DFSMSdss product to do the movement of data, including DB2 data.

DB2 datasets and volumes can be managed with this product in this manner, like any other dataset, by exploiting five constructs that are defined by the storage administrator.

These constructs are: Data Class, Management Class, Storage Class, Storage Group, and Automatic Class Selection (ACS) Routines.

The DBA should understand the meaning of these constructs in order

to communicate effectively with the storage administrator. These constructs are not the subject of this article.

Also in a DB2 Data Sharing Environment, DFSMShsm provides the best solution for managing archived log datasets to DASD.

DFSMSrmm

DFSMSrmm is the tape management system of the facility.

DFHSM

DFHSM can perform all the above mentioned activities either automatically or manually. For the purpose of this article we are going to discuss two manual commands that can be entered from a TSO terminal, either by a DBA or by storage administrator, to migrate a particular active DB2 dataset from level 0 to other levels and then to recall the migrated dataset back to level 0.

The essence of this tip is that, when DFHSM recalls a migrated dataset, it tries to consolidate the extents into one primary allocation – provided there is enough space in the SMS pool. If DFHSM can not find enough space in the SMS pool to consolidate all the extents into one primary allocation, then it will try to consolidate as many extents as it can in the primary allocation with the limited space it has.

Here is the entire procedure to consolidate 119 extents of DB2 dataset, starting from the time production support calls to the DBA for help:

1 From TSO terminal ISPF panel 3.4 or through batch IDCAM job issue a LISTCAT of the dataset in question and verify that indeed it has reached 119 extents.

2 Determine the DB2 Tablespace or Indexspace that the dataset resides in.

3 STOP the Tablespace or the Indexspace determined in step two by either using SPUFI or through DSN processor batch job.

  Verify that the Tablespace or the Indexspace has really stopped by displaying it again.

4 Enter the **Hmigrate** line command beside the dataset name cluster from TSO ISPF terminal panel 3.4. If the storage administrator has to perform this task he will issue the **Migrate** line command.

The **Migrate** command is a bit more powerful than **Hmigrate**, although for the purpose of this article both achieve the same result.

The cluster of a DB2 dataset can be identified by DSNDBC in the second-level qualifier in the dataset name.

Also note that, if the migration pool in your shop is small, DFHSM might not be able to do the migration and will inform you of its constraint.

5 Wait for DFHSM to migrate your dataset. The wait time usually depends upon the size of the dataset and the level of DFHSM migration, ie level 1 or level 2, which we talked about at the begining of this article – it should not be more than few minutes. DFHSM will prompt you with a message informing you that it has accomplished the task of migrating your dataset.

6 Exit from TSO ISPF panel 3.4 and come back to it again. Issue the **Hrecall** line command beside the now migrated dataset name. If the storage administrator has to perform this task he will issue the **Recall** line command.

The **Recall** command is a bit more powerful than **Hrecall**, although for the purpose of this article both achieve the same result.

Keep in mind that this **Recall** is a manual on-demand recall and is not related to the Automatic Recall or the lack of it as stated in two zparm values – DSN6SPRM RECALL and DSN6SPRM RECALLD.

7 Wait for DFHSM to recall your dataset. The waiting time usually depends upon the size of the dataset but it should not be more than a few minutes. In a controlled test that I have done, it took DFHSM about five minutes to recall a DB2 dataset of 2GB and consolidate its 119 extents into one primary allocation.

DFHSM will prompt you with a message informing you that it has completed the recalling task.

Whether you are in step four or in step six of this procedure, you can check the progress of the DFHSM command by issuing **Hquery** in option 6 of TSO.

8 Using LISTCAT again, verify that DFHSM has indeed consolidated the extents into one primary allocation or, depending upon the availability of space, it has consolidated the 119 extents into fewer extents.

9 START the Tablespace that you stopped in step three, either through SPUFI or through DSN Processor batch job, and verify that the Tablespace has indeed started by displaying it again.

10 Inform production support that the extent problem has been solved and go back to sleep.

11 The next morning, or at your earliest convenience, issue an SQL ALTER TABLESPACE statement to change the Priqty and Secqty of the affected Tablespace in the DB2 Catalog.

Make sure that the values you specify in the SQL ALTER statement match the values DFHSM has used during consolidation of the extents. This can be verified through LISTCAT.

Be careful how you read the units of the LISTCAT. The units of LISTCAT are in bytes and the units of SQL ALTER TABLESPACE statement are in KB.

So far you have achieved consolidation of 119 extents quickly (estimated time is between five and ten minutes) without running a DB2 Reorg!

Warning: please do not use this technique as your *modus operandi* in lieu of running the DB2 Reorg Utility. Use this procedure only in an emergency because the DB2 Reorg Utility does more than consolidation of extents. Happy snoozing!

Nicola S Nur
Senior DBA (Canada)

DB2 news

For DB2 users who want to make sure that they are year 2000-compliant, Platinum Technology has begun shipping TransCentury File Age.

The software can be used to automate the data-ageing process, and the rules-based functionality means ageing can take place according to business requirements. Another feature is an ability to automate the process of creating test databases. The software ages data, then prepares it for use in test scenarios, supporting test database creation for DB2, as well as IMS, VSAM, and sequential files for other databases.

For further information contact:
Platinum Technology, 1815 S Meyer Rd, Oakbrook Terrace, IL 60181-5241, USA.
Tel: (630) 620 5000.
Platinum Technology UK, Platinum House, N Second Street, Central Milton Keynes, Bucks, MK9 1BZ, UK.
Tel: (01908) 248400.

* * *

DB2 databases can now be linked to Web applications using the Amazon Web development tool from Intelligent Environments. The tool is integrated with SilverStream Software's Web application platform giving a product called Beyond JDBC. Amazon provides data integration from multiple relational and non-relational data sources, including 3270 screen content. It's got native connectivity options to DB2, as well as Oracle, Sybase, and SQL Server, and supports CICS, 3270 and 5250 terminal emulation, APPC, and MQSeries.

For further information contact:
Intelligent Environments, 67 S Bedford St, Burlington, MA 01803-5152, USA.
Tel: (617) 272 9700.
Intelligent Environments, 8 Windmill Business Village, Brooklands Close, Sunbury-on-Thames, TW16 7DY, UK.
Tel: (01932) 772266.

* * *

Micro Focus is to buy DB2 specialist XDB Systems, in the process getting hold of DB2 database development, maintenance, and connectivity solutions.

The companies point to industry research showing that the amount of data stored in DB2 databases will double over the next two years, while saying that IBM estimates such database servers are currently in use in 80% of Fortune 500 companies supporting 30 million users.

For further information contact:
Micro Focus, Speen Court, 7 Oxford Road, Newbury, Berks, RG14 1PB.
Tel: (01635) 32646.
Micro Focus, 2465 E Bayshore Rd, Palo Alto, CA 94303, USA.
Tel: (415) 856 4161.

* * *

Attachmate has integrated its Rally! PC-to-AS/400 connectivity product with the Nirvana query and reporting tools from Synergy Technology, targeting DB2/400 users who want to increase PC access capabilities. Rally!, which provides a link between Nirvana and the DB2/400 database, uses connectivity technology that allows the data stream to be passed via high-level APIs.

For further information contact:
Attachmate, 3617 131st Avenue SE, Bellevue, WA 98006, USA.
Tel: (206) 644 4010.
Attachmate UK, Markham House, 20 Broad St, Wokingham, Berks, RG40 1AH, UK.
Tel: (01734) 890390.