71

DB2

September 1998

In this issue

update

DB2 Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38030
From USA: 01144 1635 38030
E-mail: xephon@compuserve.com

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75077-2150
USA
Telephone: 940 455 7050

Contributions
Articles published in *DB2 Update* are paid for at the rate of £170 ($250) per 1000 words and £90 ($140) per 100 lines of code for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*.

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.

Editor
Robert Burgess

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 £245.00 in the UK; $365.00 in the USA and Canada; £251.00 in Europe; £257.00 in Australasia and Japan; and £255.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1994 issue, are available separately to subscribers for £21.00 ($31.00) each including postage.

Printed in England.

‘Transparent’ migration to DB2

INTRODUCTION

Over the years I have worked with SQL and non-SQL databases for many clients. On several occasions I have been involved in the latter stages of migration projects that are veering slightly off track – possibly because the benefit of someone’s experience was not exploited in the early stages of the project.

Many database sites carry out migrations that may, at first sight, appear to be one-off events, yet in fact many issues arise that have been encountered at other sites, even if different DBMSs are involved.

The migration of an application from one DBMS to another is a task that many sites undertake. Initially the task often appears simple, yet becomes increasingly complex (and delayed, expensive, etc) as time goes on.

There are two basic ways in which an application can be migrated from one DBMS to another:

- A wholesale rewrite.
- Some kind of ‘transparency’ solution, whereby calls to one DBMS are translated by some interface logic into calls to another.

The former option is immediately seen to be expensive, whereas the latter may initially appear to be far simpler, offering the benefits of a new DBMS while retaining much of the legacy system. By the time it becomes apparent that this may not necessarily be the case, the project may have progressed to the point of no return.

Many issues arise, whatever DBMSs are involved, although some specific solutions are available when DB2 is the DBMS being migrated to. Migration may be simpler if the DML of the original DBMS is a version of SQL, but this does not mean the migration will be wholly trouble-free.

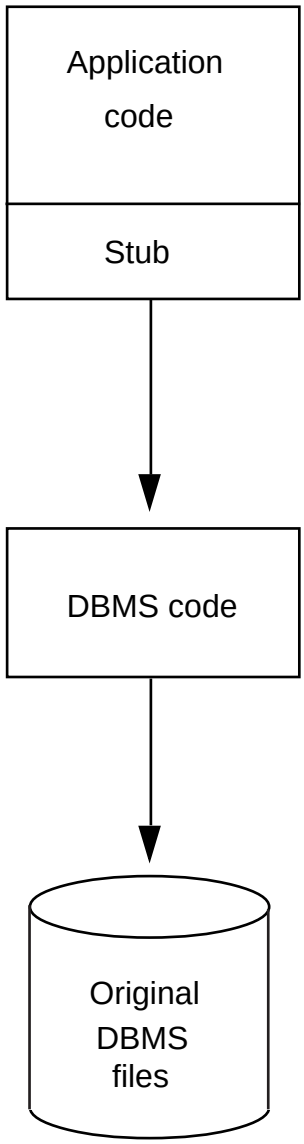

Figure 1: Using the original DBMS

THE ATTRACTIONS OF 'TRANSPARENT' MIGRATION SOLUTIONS

Figures 1 to 3 illustrate the way 'transparent' solutions work. Figure 1 shows the original DBMS application. With a 'transparent' solution, the call to the original database is intercepted at some point and translated into a call to the new database – into SQL, for DB2. There are two variations:

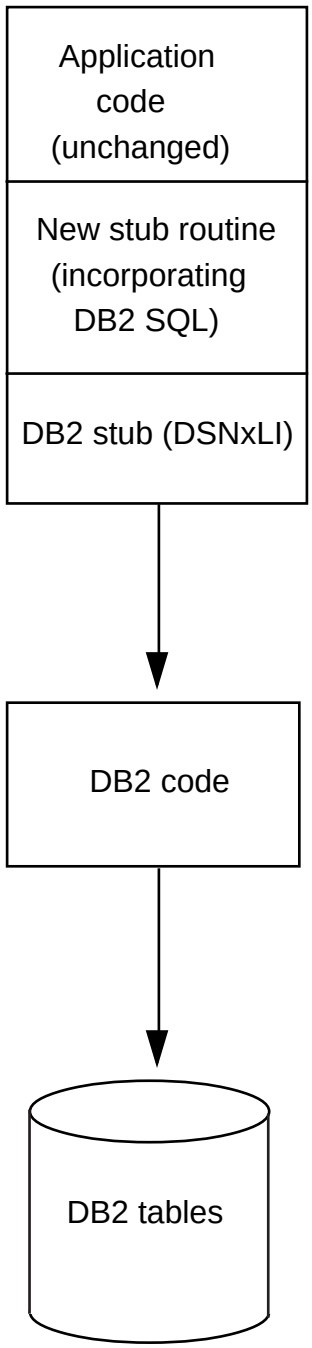

Figure 2: Using a replacement stub

- Remove all logic associated with the ‘old’ DBMS by replacing the DBMS stub linked into the application code with a special one to translate the calls. This is shown in Figure 2.
- Use an exit point, if one is available, within the existing DBMS code, to hook into file access logic and convert this to a call to the new DBMS. This is shown in Figure 3.

The use of a replacement stub has the advantage of allowing the complete removal of all code associated with the old DBMS. This may allow cancellation of licences, savings in system resources, and other benefits.

The use of an exit point may allow for simpler translation logic, or just for partial replacement of the old DBMS.

The apparent advantages of a ‘transparency’ solution from a business point of view can be overwhelming:

- The end-user view of the application does not change.
- It avoids the costs and duration of wholesale redevelopment.
- There are minimal environmental (ie TP monitor, operating system) changes.

PROBLEMS ASSOCIATED WITH ‘TRANSPARENCY’

‘Transparency’ code can be written and unit tested fairly easily. However, full simulation of a production environment is likely to be virtually impossible. To determine whether the solution could be feasible for a production system, a great deal of detailed analysis of what users need from their system must be undertaken. For instance, in many on-line applications, response times are critical – a four- or five-fold increase may not be acceptable, but may not be apparent in the early stages of testing.

Many of the problems that can arise do have potential solutions. The following section details some of the common problems I have encountered, and some ideas on how they might be avoided.

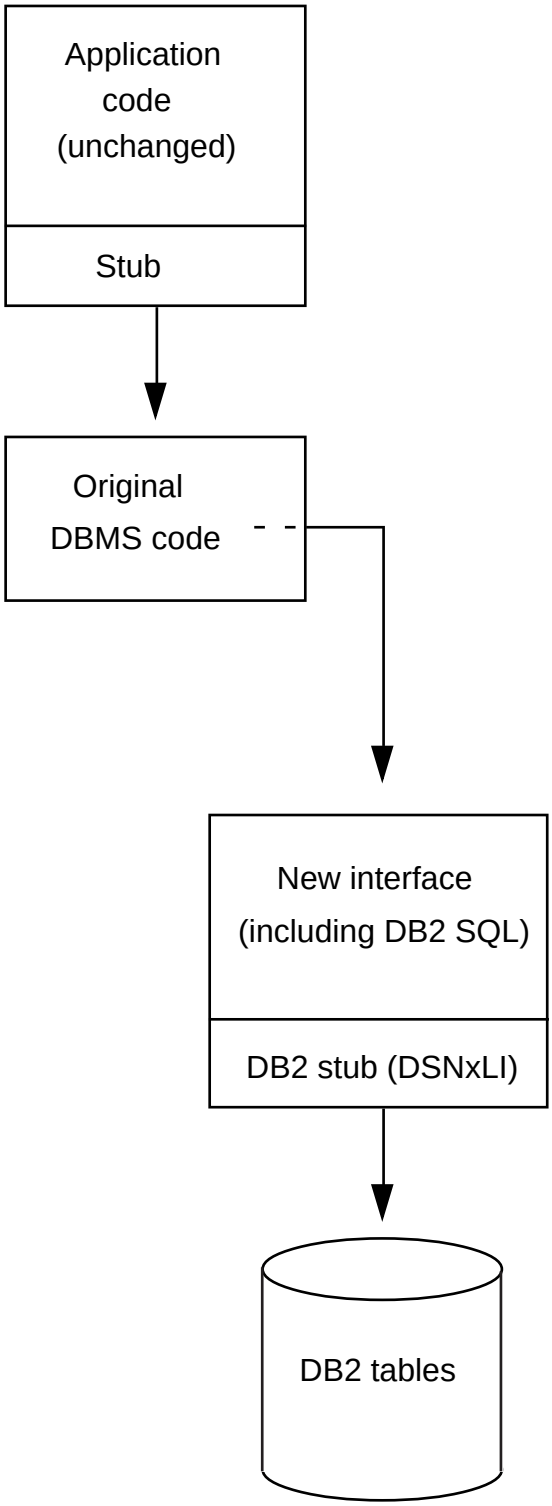

Figure 3: Using a hook in the original DBMS code

Increase in memory and CPU time for each database call

The increase in memory and CPU time is, in some ways, inevitable. Most DBMS systems are fairly efficient in processing an application call – as little code as possible is executed between the entry to the stub linked into the application code, the access to the dataset, and the return of the result to the application.

Translating the call to SQL and sending it off to DB2 is inevitably going to invoke more code. How this is done, however, can have a great bearing on the performance of the final migrated application. If many calls to CICS or MVS services are involved (loading modules perhaps), the difference between the original and the final systems is going to be rather greater than if fewer are involved.

Coding the translation logic in a high-level language may seem desirable – it will always be *possible* – but the performance overhead may be severe. What can be achieved in a few KB of Assembler load module may require a hundred or more KB when generated from COBOL or PL/I. If this has to be loaded for every DBMS call, the effect could be very significant.

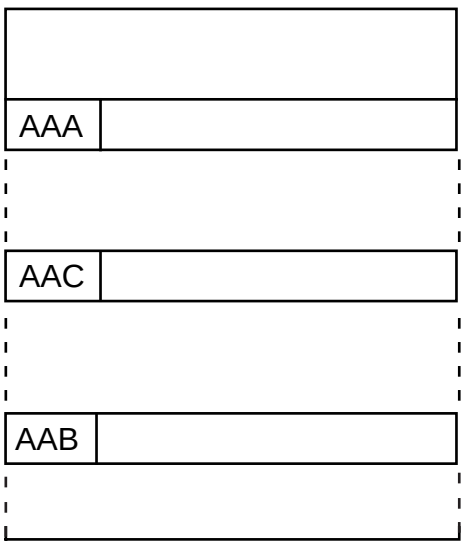

Figure 4: Hashed placement of records within a dataset

Apparently random variation in elapsed time for each DBMS call

Whereas the increase in memory and CPU time for each database call represents a fairly predictable performance overhead, there is also an effect that may arise occasionally or even frequently – depending, perhaps, upon the overall level of activity on the system or possibly even what functions are being carried out within the application. This arises when assumptions implicit in the original application design

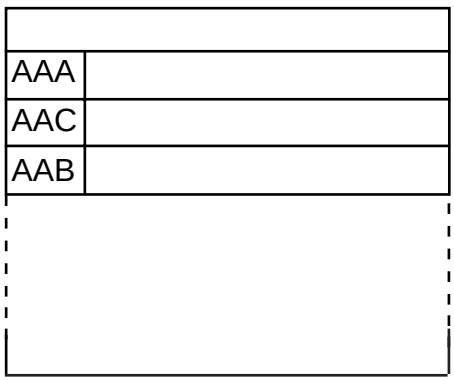

Figure 5: Clustered records placed on tablespace

become something of a liability with the new DBMS. This often occurs when migrating from a DBMS with a hashed record positioning algorithm, as shown in Figure 4, to one (such as DB2) which involves clustering (Figure 5). If the same fields are used as primary (locator) keys, records which, under the original DBMS, were located on physically separate parts of the dataset may, under DB2, be stored in adjacent locations. If page-level locking is in effect, one thread accessing one record may prevent another thread accessing the now-adjacent record. This can particularly arise when the locator key is in some way date- or time-dependent.

Enforced single threading of applications

A partial solution to both of the problems outlined above is to enforce single threading of parts of the application that encounter these

problems. In the days when each system had only a single CPU this may have been almost acceptable, but nowadays any single threading must surely be considered undesirable. It may be that past solutions to the types of problem outlined have given rise to enforced single threading. It may be worth reviewing any application areas with this feature to see whether, with the features in later releases (eg DB2 V4 LOCKSIZE(ROW), ISOLATION(UR), etc), the single threading bottleneck can be overcome.

Hidden functions in original DBMS

If the application being migrated used any hidden functions in the original DBMS, these will, in some way, have to be handled within the interface code. Typical examples are to extract schema information relating to data being accessed, or processing of error codes and messages.

SQL code can be written to do at least some of this, or, if it is deemed acceptable to have the interface dependent upon the specific application, it may be possible to hardcode appropriate logic into the interface. This can be especially important if the original application was not designed to issue the calls to the DBMS itself, but used some middleware for this – ie code exists between the application and the original DBMS stub that is not entirely known or under the control of the client undertaking the migration, such as when 4GLs are involved.

A similar issue may arise when non-relational aspects of a database are exploited by an application, such as sequencing of records other than by a key, or sub-definition of fields within records. (In DB2 it is not possible to have the same data values identified by more than one column name.)

Different interpretation of essential concepts

Finally, it is dangerous to make any assumptions about apparently basic concepts of database processing. DB2 and IMS conform closely to the principle of the start and end of the logical work being determined by the transaction manager (CICS or IMS). Other DBMSs do not – to the extent that they will hold locks on records after the end

of a CICS transaction, and may not even trigger an immediate release of resources on task abend. Whilst this may only affect very small parts of an application, it can be very difficult to resolve if assumptions implicit in the original application design require that the DBMS should behave as the original one really did.

SOME LESSONS

There are several lessons that can be learnt:

- Consider precisely what the priorities are for the interface code – can you afford the luxury of sticking to 'standards' if this means, say, sacrificing the performance advantages of coding in Assembler?

  Further to this, the essence of the interface code must be simplicity in execution – excessive calls to TP monitor or MVS resources should be avoided.

- If the installation of a later version of DB2 is needed to take advantage of potentially useful DB2 features, this should be considered a necessary prerequisite for the project.

- Learn everything you can about the way your application uses the original DBMS. For example, does it use any non-standard facilities? Do you have sufficient information to be able to *completely* replicate them using the new DBMS?

I hope these ideas prove to be of use and interest. I am sure that many other practitioners have encountered the kind of issues I have outlined here, and I am sure their thoughts would make for interesting reading in future issues of *DB2 Update*.

Phil Button
Consultant (UK)

Analysing and tuning DB2 buffer pools, revisited

This article describes the additional coding required to handle multiple character DB2 recognition-ids in the EXEC presented in *Analysing and tuning DB2 buffer pools* published in *DB2 Update*, Issue 50, December 1996.

There are three additional coding requirements to the original EXEC to allow it to handle multiple-character DB2 recognition-ids. The original code assumed that the recognition character was a single character, and used specific positioning techniques to get the buffer pool values from the DB2 command. The code below will allow you to use any number of characters as the recognition value.

The code below makes no reference to the comments made by Joel Goldstein in *Buffer pool tuning and resizing*, *DB2 Update*, Issue 59, September 1997, as to how the hit ratios should be calculated.

The procedure is as follows. Firstly, change the line which reads:

```
X = OUTTRAP('L.','*','NOCONCAT')
```

to read:

```
X = OUTTRAP('LIN.','*','NOCONCAT')
```

Next, after the 'CALC= 'N'' line, add the following:

```
XL = Ø
DO J = 1 TO LIN.Ø
 IF(SUBSTR(SUBWORD(LIN.J,1,),1,4) = 'DSNB') THEN DO
  XL = XL + 1
   L.XL = LIN.J
    END /* IF(SUBSTR(SUBWORD(LIN.J,1,),1,4) = 'DSNB') THEN DO */
    ELSE DO
   L.XL = L.XL LIN.J
  END /* IF(SUBSTR(SUBWORD(LIN.J,1,),1,4) = 'DSNB') THEN DO */
END /* DO J = 1 TO L.Ø */
L.Ø = XL
```

Finally, replace the lines starting with 'BUFNAME:' to the line before 'ALTER:' with:

```
BUFNAME:
       CALL AFF
      PARSE VAR L.J 'USE COUNT' USEVBP .
RETURN(Ø)
```

```
VIRBUF:
   CALL AFF
   PARSE VAR L.J 'VIRTUAL BUFFERPOOL SIZE =' SIZEVBP .
   PARSE VAR L.J 'ALLOCATED       =' ALLOCVBP  .
   PARSE VAR L.J 'TO BE DELETED   =' BEDELVBP  .
   PARSE VAR L.J 'IN-USE/UPDATED  =' USEUPDVBP .
   RETURN(Ø)
HPCAST:
   CALL AFF
   PARSE VAR L.J 'HIPERPOOL SIZE =' SIZEHP    .
   PARSE VAR L.J 'CASTOUT =' CASTOUT   .
   PARSE VAR L.J 'ALLOCATED       =' ALLOCHP   .
   PARSE VAR L.J 'TO BE DELETED   =' BEDELHP   .
   PARSE VAR L.J 'BACKED BY ES    =' BACKESHP  .
   RETURN(Ø)
THRESHOLD:
   PARSE VAR L.J 'VP SEQUENTIAL         =' VPSTHHP .
   PARSE VAR L.J 'HP SEQUENTIAL         =' HPSTHHP .
   PARSE VAR L.J 'DEFERRED WRITE        =' DEFWRT  .
   PARSE VAR L.J 'VERTICAL DEFERRED WRT =' VDFWRT  .
   PARSE VAR L.J 'PARALLEL SEQUENTIAL   =' PARSEQ  .
   RETURN(Ø)
STATS:
   CALL AFF
   PARSE VAR L.J 'SINCE ' STAT
   RETURN(Ø)
GETPAGE:
   CALL AFF
   PARSE VAR L.J 'RANDOM GETPAGE    =' RDNGET  .
   PARSE VAR L.J 'SYNC READ I/O (R) =' SRIOR   .
   PARSE VAR L.J 'SEQ.   GETPAGE    =' SEQGET  .
   PARSE VAR L.J 'SYNC READ I/O (S) =' SRIOS   .
   PARSE VAR L.J 'DMTH HIT          =' DMTHHIT .
   RETURN(Ø)
SPREFETCH:
SPREFETCH:
   CALL AFF
   PARSE VAR L.J 'REQUESTS        =' SQPRRQ  .
   PARSE VAR L.J 'PREFETCH I/O    =' SQPRIO  .
   PARSE VAR L.J 'PAGES READ      =' SQPRPG  .
   RETURN(Ø)
LPREFETCH:
   CALL AFF
   PARSE VAR L.J 'REQUESTS        =' LSPRRQ  .
   PARSE VAR L.J 'PREFETCH I/O    =' LSPRIO  .
   PARSE VAR L.J 'PAGES READ      =' LSPRPG  .
   RETURN(Ø)
DPREFETCH:
   CALL AFF
   PARSE VAR L.J 'REQUESTS        =' DYPRRQ  .
   PARSE VAR L.J 'PREFETCH I/O    =' DYPRIO  .
```

```
   PARSE VAR L.J 'PAGES READ      =' DYPRPG  .
   RETURN(Ø)
PREFDIS:
   CALL AFF
PARSE VAR L.J 'NO BUFFER       =' PRDIBUF .
   PARSE VAR L.J 'NO READ ENGINE  =' PRDIENG .
   RETURN(Ø)
AFF:
  IF AFF = 'Y' THEN SAY L.J
  RETURN(Ø)
PAGES:
   CALL AFF
   PARSE VAR L.J 'SYS PAGE UPDATES  =' SYSPGUP  .
   PARSE VAR L.J 'SYS PAGES WRITTEN =' SYSPGWR  .
   PARSE VAR L.J 'ASYNC WRITE I/O   =' ASYNCWIO .
   PARSE VAR L.J 'SYNC WRITE I/O    =' SYNCWIO .
   RETURN(Ø)
HIT:
   CALL AFF
   PARSE VAR L.J 'DWT HIT           =' DWTHIT .
   PARSE VAR L.J 'VERTICAL DWT HIT  =' VDWTHIT .
   PARSE VAR L.J 'NO WRITE ENGINE   =' NOWENG  .
   RETURN(Ø)
HPNUADMF:
   CALL AFF
   PARSE VAR L.J 'SYNC HP READS   =' SYNCHPR  .
   PARSE VAR L.J 'SYNC HP WRITES  =' SYNCHPW  .
   PARSE VAR L.J 'ASYNC HP READS  =' ASYNCHPR .
   PARSE VAR L.J 'ASYNC HP WRITES =' ASYNCHPW .
   PARSE VAR L.J 'READ FAILURES   =' READFAIL .
   PARSE VAR L.J 'WRITE FAILURES  =' WRITFAIL .
   RETURN(Ø)
HPUADMF:
   CALL AFF
   PARSE VAR L.J 'HP READS        =' HPREAD  .
   PARSE VAR L.J 'HP WRITES       =' HPWRITE .
   PARSE VAR L.J 'READ FAILURES   =' HPRFAIL .
   PARSE VAR L.J 'WRITE FAILURES  =' HPWFAIL .
   RETURN(Ø)
PARACT:
   CALL AFF
   PARSE VAR L.J 'PARALLEL REQUEST =' PARREQ .
   PARSE VAR L.J 'DEGRADED PARALLEL=' DEGPAR .
   RETURN(Ø)
HPNAME:
   CALL AFF
   X = INDEX(L.J,'HIPERSPACE NAME(S) - ')
   IF X > Ø THEN HPNAME = SUBSTR(L.J,X+21)
   RETURN(Ø)
```


Simulating a production environment

I'm sure it must have happened at your DB2 site – after being tested and accepted, a certain application in DB2 had a tremendous impact when moved to production. The main influences on DB2 optimizer's choice of access path are the number of rows per table, the index's cardinality, etc.

You may experience some of the following difficulties when trying to simulate a production environment:

- It's time-consuming to input sufficient testing data.
- It is difficult to ascertain at which point the Optimizer changes the access path.
- It takes time to update the catalog statistics in the testing environment.
- The sudden increase from a low volume to a high volume of data.

From personal experience, I think it is worth considering two different types of environment:

- The modified application.
- The new application.

In both cases, I use two VS COBOL II programs (I am currently running in a MVS 4.3 with JES2 4.3, DB2 3.1, and VS COBOL II 3.2 environment).

PROGRAMS

The PCATV3EX program extracts all catalog information that influences the Optimizer's access path selection for any environment by specifying a database. This program produces six sequential files described below:

- SYSTBLS – Output with some columns of SYSIBM. SYSTABLES.

- SYSTBLSP – Output with some columns of SYSIBM. SYSTABLESPACE.
- SYSINDXS – Output with some columns of SYSIBM. SYSINDEXES.
- SYSCOLMS – Output with some columns of SYSIBM. SYSCOLUMNS.
- SYSCOLDT – Output with some columns of SYSIBM. SYSCOLDIST.
- REPOUT – Output when the program fails.

PCATV3UP updates all catalog information for any environment to influence the Optimizer's access path selection for a specified database. This program uses the sequential files produced by the previous program to execute.

For the first type of environment, this concludes the scenario. For the second type of environment, I have five REXX EXECs that must be executed between the two COBOL programs, one for each file produced by the first program. Each REXX EXEC searches its respective file produced by program PCATV3EX, specifying all entries one by one, or one specific entry that we choose, in order to produce the respective output files with the new values for the fields to be updated by program PCATV3UP.

PCATV3EX

```
 IDENTIFICATION DIVISION.
       PROGRAM-ID. PCATV3EX.
       AUTHOR. IOLANDA LOPES.
      *
      *   ---------------------------------
      *   -   *****  PROGRAM THAT SELECTS STATISTICS THAT  ******   -
      *   -   *****  INFLUENCE THE ACCESS PATH OF PROGRAMS ******   -
      *   -   *****  FROM A SPECIFIC DB2 CATALOG TABLES    ******   -
      *   ---------------------------------
      *
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. IBM.
       OBJECT-COMPUTER. IBM.
```

```
 SPECIAL-NAMES.
     CONSOLE IS CONSOLA
     DECIMAL-POINT IS COMMA.
*
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
*
     SELECT  SYSTBLS  ASSIGN TO SYSTBLS
              FILE STATUS IS  FILE-STATUS.

     SELECT  SYSTBLSP ASSIGN TO SYSTBLSP
              FILE STATUS IS  FILE-STATUS.

     SELECT  SYSINDXS ASSIGN TO SYSINDXS
              FILE STATUS IS  FILE-STATUS.

     SELECT  SYSCOLMS ASSIGN TO SYSCOLMS
              FILE STATUS IS  FILE-STATUS.

     SELECT  SYSCOLDT ASSIGN TO SYSCOLDT
              FILE STATUS IS  FILE-STATUS.

     SELECT  REPOUT   ASSIGN TO REPOUT.
*
 DATA DIVISION.
 FILE SECTION.
*
 FD  SYSTBLS
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS R-SYSTBLS.

 Ø1  R-SYSTBLS          PIC X(4Ø).
*
 FD  SYSTBLSP
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS R-SYSTBLSP.

 Ø1  R-SYSTBLSP         PIC X(17).
*
 FD  SYSINDXS
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS R-SYSINDXS.

 Ø1  R-SYSINDXS         PIC X(71).
*
 FD  SYSCOLMS
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS R-SYSCOLMS.

 Ø1  R-SYSCOLMS         PIC X(69).
```

```
*
 FD  SYSCOLDT
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS R-SYSCOLDT.

 Ø1  R-SYSCOLDT         PIC X(294).
*
 FD  REPOUT
     RECORD CONTAINS 12Ø CHARACTERS
     LABEL RECORDS ARE OMITTED
     DATA RECORD IS REPREC.
 Ø1  REPREC               PIC X(12Ø).
*
*
 WORKING-STORAGE SECTION.
*
*
 Ø1  FILE-STATUS        PIC 99     VALUE ØØ.
*
 Ø1  ERROR-MESSAGE.
     Ø2  ERROR-LEN      PIC S9(4)  COMP VALUE +96Ø.
     Ø2  ERROR-TEXT     PIC X(12Ø) OCCURS 8 TIMES
                                   INDEXED BY ERROR-INDEX.
 77  ERROR-TEXT-LEN     PIC S9(4)  COMP VALUE +12Ø.
*
*————————————————————————————
*    DB2 ERROR MESSAGES
*————————————————————————————
*
 Ø1  MSG2.
     Ø2 NAME-TABLE            PIC X(5).
     Ø2 FILLER                PIC X(13) VALUE ' SQLCOD-ERRO'.
     Ø2 ERROR-CODE            PIC -9(9).
*
*————————————————————————————
*    WORKAREAS                                                  -
*————————————————————————————
*
*
 Ø1  W-SYSTBLS.
     Ø5  W-TBL-NAME             PIC X(18).
     Ø5  W-TBL-CARD             PIC X(9).
     Ø5  W-TBL-NPAGES           PIC X(9).
     Ø5  W-TBL-PCTRCOMP         PIC X(4).
*
 Ø1  W-TBL-CARD-NUM             PIC 9(9).
 Ø1  W-TBL-CARD-NUM-R      REDEFINES
     W-TBL-CARD-NUM             PIC S9(9) COMP SYNC.
*
 Ø1  W-TBL-NPAGES-NUM           PIC 9(9).
```

```
 01  W-TBL-NPAGES-NUM-R      REDEFINES
     W-TBL-NPAGES-NUM           PIC S9(9) COMP SYNC.
*
 01  W-TBL-PCTRCOMP-NUM         PIC 9(4).
 01  W-TBL-PCTRCOMP-NUM-R   REDEFINES
     W-TBL-PCTRCOMP-NUM         PIC S9(4) COMP SYNC.
*
 01  W-SYSTBLSP.
     05  W-TBLSP-NAME           PIC X(8).
     05  W-TBLSP-NACTIVE        PIC X(9).
*
 01  W-TBLSP-NACTIVE-NUM        PIC 9(9).
 01  W-TBLSP-NACTIVE-NUM-R REDEFINES
     W-TBLSP-NACTIVE-NUM        PIC S9(9) COMP SYNC.

 01  W-SYSINDXS.
     05  W-INDXS-TBNAME         PIC X(18).
     05  W-INDXS-NAME           PIC X(18).
     05  W-CLUSTERRATIO         PIC X(4).
     05  W-FIRSTKEY-CARD        PIC X(9).
     05  W-FULLKEY-CARD         PIC X(9).
     05  W-NLEAF                PIC X(9).
     05  W-NLEVELS              PIC X(4).
*
 01  W-CLUSTERRATIO-NUM         PIC 9(4).
 01  W-CLUSTERRAT-NUM-R  REDEFINES
     W-CLUSTERRATIO-NUM         PIC S9(4) COMP SYNC.
*
 01  W-FIRSTKEY-CARD-NUM        PIC 9(9).
 01  W-FIRSTKEY-NUM-R    REDEFINES
     W-FIRSTKEY-CARD-NUM        PIC S9(9) COMP SYNC.
*
 01  W-FULLKEY-CARD-NUM         PIC 9(9).
 01  W-FULLKEY-NUM-R     REDEFINES
     W-FULLKEY-CARD-NUM         PIC S9(9) COMP SYNC.
*
 01  W-NLEAF-NUM                PIC 9(9).
 01  W-NLEAF-NUM-R       REDEFINES
     W-NLEAF-NUM                PIC S9(9) COMP SYNC.
*
 01  W-NLEVELS-NUM              PIC 9(4).
 01  W-NLEVELS-NUM-R     REDEFINES
     W-NLEVELS-NUM              PIC S9(4) COMP SYNC.
*
*
 01  W-SYSCOLMS.
     05  W-COLMS-NAME           PIC X(18).
     05  W-COLMS-TBNAME         PIC X(18).
     05  W-COLCARD              PIC X(9).
     05  W-LOW2KEY              PIC X(8).
```

```
     05  W-HIGH2KEY             PIC X(8).
     05  W-COLTYPE              PIC X(8).
*
 01  W-COLCARD-NUM              PIC 9(9).
 01  W-COLCARD-NUM-R     REDEFINES
     W-COLCARD-NUM               PIC S9(9) COMP SYNC.
*
*
 01  W-SYSCOLDT.
     05  W-COLDT-NAME           PIC X(18).
     05  W-COLDT-TBNAME         PIC X(18).
     05  W-COLVALUE             PIC X(254).
     05  W-FREQUENC             PIC X(4).
*
 01  W-FREQUENC-NUM             PIC 9(4).
 01  W-FREQUENC-NUM-R     REDEFINES
     W-FREQUENC-NUM              PIC S9(4) COMP SYNC.
*
*
 01  WS-SWITCHES.
     05  END-PROCESS            PIC 9(1) VALUE 0.
*
      01  COUNT-TBL-UPD              PIC 9(9)     VALUE 0.
*
*
 01  W-PARM.
     05  W-PARM-DBNAME          PIC X(8) VALUE SPACES.
     05  W-PARM-TBOWNER         PIC X(8) VALUE SPACES.

*------------------------------------
*    INCLUDE SQL COMMUNICATION AREA                                -
*------------------------------------
*
     EXEC SQL
              INCLUDE SQLCA
     END-EXEC.
*
*
 PROCEDURE DIVISION.
*
*
        ACCEPT W-PARM FROM SYSIN.
        DISPLAY 'DBNAME' W-PARM-DBNAME.
        DISPLAY 'TBOWNER' W-PARM-TBOWNER.
*
*
        EXEC SQL DECLARE FETCH_SYSTBLS CURSOR FOR
            SELECT NAME, CARD, NPAGES, PCTROWCOMP
              FROM SYSIBM.SYSTABLES
            WHERE DBNAME  = :W-PARM-DBNAME
```

```
           END-EXEC.
      *
           EXEC SQL DECLARE FETCH_SYSTBLSP CURSOR FOR
               SELECT NAME, NACTIVE
                 FROM SYSIBM.SYSTABLESPACE
               WHERE DBNAME = :W-PARM-DBNAME
           END-EXEC.
      *
           EXEC SQL DECLARE FETCH_SYSINDXS CURSOR FOR
               SELECT TBNAME, NAME, FIRSTKEYCARD,
                      FULLKEYCARD, NLEAF, NLEVELS, CLUSTERRATIO
                 FROM SYSIBM.SYSINDEXES
               WHERE DBNAME = :W-PARM-DBNAME
           END-EXEC.
      *
           EXEC SQL DECLARE FETCH_SYSCOLMS CURSOR FOR
               SELECT TBNAME, NAME, COLCARD,
                      LOW2KEY, HIGH2KEY, COLTYPE
                 FROM  SYSIBM.SYSCOLUMNS
               WHERE TBCREATOR = :W-PARM-TBOWNER
           END-EXEC.
      *
           EXEC SQL DECLARE FETCH_SYSCOLDT CURSOR FOR
               SELECT TBNAME, NAME, COLVALUE, FREQUENCY
                 FROM  SYSIBM.SYSCOLDIST
               WHERE TBOWNER = :W-PARM-TBOWNER
               ORDER BY TBNAME, NAME
           END-EXEC.
      *
          EXEC SQL
             WHENEVER SQLERROR GO TO X100-DBERROR-DB2
          END-EXEC.
      *
      *
          EXEC SQL
             WHENEVER SQLWARNING GO TO X100-DBERROR-DB2
          END-EXEC.
      *
      *
          EXEC SQL
             WHENEVER NOT FOUND CONTINUE
          END-EXEC.
      *-------------------------------
      *    GENERAL STRUCTURE OF PROGRAM                               -
      *-------------------------------
      *
       0000-MAIN.
      *
          PERFORM A100-BEGIN
             THRU A100-BEGIN-EXIT.
```

```
      *
           PERFORM B100-INITIALIZE
              THRU B100-INITIALIZE-EXIT.
      *
           PERFORM C100-OPEN-SYSTBLS
              THRU C100-OPEN-SYSTBLS-EXIT.
      *
           PERFORM C200-OPEN-SYSTBLSP
              THRU C200-OPEN-SYSTBLSP-EXIT.
      *
           PERFORM C300-OPEN-SYSINDXS
              THRU C300-OPEN-SYSINDXS-EXIT.
      *
           PERFORM C400-OPEN-SYSCOLMS
              THRU C400-OPEN-SYSCOLMS-EXIT.
      *
           PERFORM C500-OPEN-SYSCOLDT
              THRU C500-OPEN-SYSCOLDT-EXIT.
      *
           PERFORM Z100-END
              THRU Z100-END-EXIT.

           STOP RUN.
      *-------------------------------
      *    OPENS FILES
      *-------------------------------
      *
       A100-BEGIN.
      *
           OPEN OUTPUT SYSTBLS
                       SYSTBLSP
                       SYSINDXS
                       SYSCOLMS
                       SYSCOLDT
                       REPOUT.
      *
       A100-BEGIN-EXIT.
           EXIT.

*-----------------------------------
      *    INITIALIZES VARIABLE FIELDS
      *-------------------------------
      *
       B100-INITIALIZE.
      *
           MOVE SPACES TO W-SYSTBLS
                          W-SYSTBLSP
                          W-SYSINDXS
                          W-SYSCOLMS
                          W-SYSCOLDT.
```

```
      *
           MOVE ZEROS TO W-TBL-CARD-NUM
                         W-TBL-NPAGES-NUM
                         W-TBL-PCTRCOMP-NUM
                         W-TBLSP-NACTIVE-NUM
                         W-CLUSTERRATIO-NUM
                         W-FIRSTKEY-CARD-NUM
                         W-FULLKEY-CARD-NUM
                         W-NLEAF-NUM
                         W-NLEVELS-NUM
                         W-COLCARD-NUM
                         W-FREQUENC-NUM.
      *
       B100-INITIALIZE-EXIT.
           EXIT.
      *
      *-------------------------------------
      *      SCANS THRU SYSIBM.SYSTABLES
      *-------------------------------------
      *
       C100-OPEN-SYSTBLS.
      *
           EXEC SQL
               OPEN FETCH_SYSTBLS
           END-EXEC.
      *
           IF SQLCODE = 0
               PERFORM C110-FETCH-SYSTBLS
                 UNTIL SQLCODE = 100
           ELSE
              IF SQLCODE NOT = +100
                 PERFORM X100-DBERROR-DB2
                    THRU X100-DBERROR-DB2-EXIT.
      *
           IF SQLCODE = 100
              EXEC SQL
                  CLOSE FETCH_SYSTBLS
              END-EXEC.
      *
       C100-OPEN-SYSTBLS-EXIT.
           EXIT.
      *

      C110-FETCH-SYSTBLS.
           *
                EXEC SQL
                   FETCH  FETCH_SYSTBLS
                       INTO  :W-TBL-NAME,
                             :W-TBL-CARD-NUM-R,
                             :W-TBL-NPAGES-NUM-R,
```

```
                    :W-TBL-PCTRCOMP-NUM-R
     END-EXEC.
*
     IF SQLCODE = Ø
        MOVE W-TBL-CARD-NUM     TO W-TBL-CARD
        MOVE W-TBL-NPAGES-NUM   TO W-TBL-NPAGES
        MOVE W-TBL-PCTRCOMP-NUM TO W-TBL-PCTRCOMP
        WRITE R-SYSTBLS FROM W-SYSTBLS
     ELSE
        IF SQLCODE NOT = +1ØØ
           PERFORM X1ØØ-DBERROR-DB2
              THRU X1ØØ-DBERROR-DB2-EXIT.
*
 C11Ø-FETCH-SYSTBLS-EXIT.
     EXIT.
*———————————————————————————
*      SCANS THRU SYSIBM.SYSTABLESPACE
*———————————————————————————
 C2ØØ-OPEN-SYSTBLSP.
*
     EXEC SQL
         OPEN FETCH_SYSTBLSP
     END-EXEC.
*
     IF SQLCODE = Ø
         PERFORM C21Ø-FETCH-SYSTBLSP
           UNTIL SQLCODE = 1ØØ
     ELSE
        IF SQLCODE NOT = +1ØØ
           PERFORM X1ØØ-DBERROR-DB2
              THRU X1ØØ-DBERROR-DB2-EXIT.
*
     IF SQLCODE = 1ØØ
        EXEC SQL
            CLOSE FETCH_SYSTBLSP
        END-EXEC.
*
 C2ØØ-OPEN-SYSTBLSP-EXIT.
     EXIT.
 C21Ø-FETCH-SYSTBLSP.
*
     EXEC SQL
         FETCH  FETCH_SYSTBLSP
            INTO  :W-TBLSP-NAME,
                  :W-TBLSP-NACTIVE-NUM-R
     END-EXEC.
*
     IF SQLCODE = Ø
        MOVE W-TBLSP-NACTIVE-NUM TO W-TBLSP-NACTIVE
        WRITE R-SYSTBLSP FROM W-SYSTBLSP
```

```
     ELSE
        IF SQLCODE NOT = +100
           PERFORM X100-DBERROR-DB2
              THRU X100-DBERROR-DB2-EXIT.
*
 C210-FETCH-SYSTBLSP-EXIT.
     EXIT.
*-----------------------------------
*      SCANS THRU SYSIBM.SYSINDEXES
*-----------------------------------
*
 C300-OPEN-SYSINDXS.
*
     EXEC SQL
         OPEN FETCH_SYSINDXS
     END-EXEC.
*
     IF SQLCODE = 0
         PERFORM C310-FETCH-SYSINDXS
           UNTIL SQLCODE = 100
     ELSE
        IF SQLCODE NOT = +100
           PERFORM X100-DBERROR-DB2
              THRU X100-DBERROR-DB2-EXIT.
*
     IF SQLCODE = 100
        EXEC SQL
            CLOSE FETCH_SYSINDXS
        END-EXEC.
*
 C300-OPEN-SYSINDXS-EXIT.
     EXIT.
 C310-FETCH-SYSINDXS.
*
     EXEC SQL
        FETCH  FETCH_SYSINDXS
            INTO  :W-INDXS-TBNAME,
                  :W-INDXS-NAME,
                  :W-FIRSTKEY-NUM-R,
                  :W-FULLKEY-NUM-R,
                  :W-NLEAF-NUM-R,
                  :W-NLEVELS-NUM-R,
                  :W-CLUSTERRAT-NUM-R
     END-EXEC.
*
     IF SQLCODE = 0
        MOVE W-CLUSTERRATIO-NUM  TO W-CLUSTERRATIO
        MOVE W-FIRSTKEY-CARD-NUM TO W-FIRSTKEY-CARD
        MOVE W-FULLKEY-CARD-NUM  TO W-FULLKEY-CARD
        MOVE W-NLEAF-NUM         TO W-NLEAF
```

```
               MOVE W-NLEVELS-NUM       TO W-NLEVELS
               WRITE R-SYSINDXS FROM W-SYSINDXS
            ELSE
               IF SQLCODE NOT = +100Ø
                  MOVE W-TBL-NAME TO NAME-TABLE
                  PERFORM X1ØØ-DBERROR-DB2
                     THRU X1ØØ-DBERROR-DB2-EXIT.
      *
       C31Ø-FETCH-SYSINDXS-EXIT.
           EXIT.
      *

*-------------------------------------
      *      SCANS THRU SYSIBM.SYSCOLUMNS
      *-------------------------------------
      *
       C4ØØ-OPEN-SYSCOLMS.
      *
           EXEC SQL
               OPEN FETCH_SYSCOLMS
           END-EXEC.
      *
           IF SQLCODE = Ø
               PERFORM C41Ø-FETCH-SYSCOLMS
                 UNTIL SQLCODE = 1ØØ
           ELSE
              IF SQLCODE NOT = +100Ø
                 PERFORM X1ØØ-DBERROR-DB2
                    THRU X1ØØ-DBERROR-DB2-EXIT.
      *
           IF SQLCODE = 1ØØ
              EXEC SQL
                  CLOSE FETCH_SYSCOLMS
              END-EXEC.
      *
       C4ØØ-OPEN-SYSCOLMS-EXIT.
           EXIT.
       C41Ø-FETCH-SYSCOLMS.
      *
           EXEC SQL
               FETCH  FETCH_SYSCOLMS
                  INTO  :W-COLMS-NAME,
                        :W-COLMS-TBNAME,
                        :W-COLCARD-NUM-R,
                        :W-LOW2KEY,
                        :W-HIGH2KEY,
                        :W-COLTYPE
           END-EXEC.
      *
           IF SQLCODE = Ø
```

```
               MOVE  W-COLCARD-NUM TO W-COLCARD
               WRITE R-SYSCOLMS FROM W-SYSCOLMS
            ELSE
               MOVE W-COLMS-TBNAME TO NAME-TABLE
               IF SQLCODE NOT = +100
                  PERFORM X100-DBERROR-DB2
                     THRU X100-DBERROR-DB2-EXIT.
      *
       C410-FETCH-SYSCOLMS-EXIT.
            EXIT.
      *

*-------------------------------------
      *      SCANS THRU SYSIBM.SYSCOLDIST
      *-------------------------------------
      *
       C500-OPEN-SYSCOLDT.
      *
            EXEC SQL
                OPEN FETCH_SYSCOLDT
            END-EXEC.
      *
            IF SQLCODE = 0
                PERFORM C510-FETCH-SYSCOLDT
                  UNTIL SQLCODE = 100
            ELSE
               IF SQLCODE NOT = +100
                  PERFORM X100-DBERROR-DB2
                     THRU X100-DBERROR-DB2-EXIT.
      *
            IF SQLCODE = 100
               EXEC SQL
                   CLOSE  FETCH_SYSCOLDT
               END-EXEC.
      *
       C500-OPEN-SYSCOLDT-EXIT.
            EXIT.
       C510-FETCH-SYSCOLDT.
      *
            EXEC SQL
                FETCH  FETCH_SYSCOLDT
                   INTO  :W-COLDT-NAME,
                         :W-COLDT-TBNAME,
                         :W-COLVALUE,
                         :W-FREQUENC-NUM-R
            END-EXEC.
      *
            IF SQLCODE = 0
               MOVE W-FREQUENC-NUM TO W-FREQUENC
               WRITE R-SYSCOLDT FROM W-SYSCOLDT
```

```
           ELSE
              MOVE W-COLDT-TBNAME TO NAME-TABLE
              IF SQLCODE NOT = +100
                 PERFORM X100-DBERROR-DB2
                    THRU X100-DBERROR-DB2-EXIT.
      *
       C510-FETCH-SYSCOLDT-EXIT.
           EXIT.
      *
      *

*-------------------------------------
      *    CALL DB2 ERROR ROUTINE
      *-------------------------------------
      *
       X100-DBERROR-DB2.
      *
           MOVE SQLCODE TO ERROR-CODE.
           DISPLAY MSG2.

           PERFORM X200-CALL-MODULE
              THRU X200-CALL-MODULE-EXIT

           PERFORM Z100-END
              THRU Z100-END-EXIT.
      *
       X100-DBERROR-DB2-EXIT.
           EXIT.
      *
       X200-CALL-MODULE.
      *
           CALL 'DSNTIAR' USING  SQLCA  ERROR-MESSAGE  ERROR-TEXT-LEN.
      *
           IF RETURN-CODE = ZERO
               PERFORM ERROR-PRINT
                  VARYING ERROR-INDEX FROM 1 BY 1
                    UNTIL ERROR-INDEX > 8.
            GO X200-CALL-MODULE-EXIT.
      *
      ****************************************************************
      * PRINT MESSAGE TEXT                                           *
      ****************************************************************
      *
       ERROR-PRINT.
      *
           WRITE REPREC FROM ERROR-TEXT (ERROR-INDEX)
              AFTER ADVANCING 1 LINE.
      *
       X200-CALL-MODULE-EXIT.
           EXIT.
```

```
      *
      ****************************************************************
      * CLOSE FILES AND END PROGRAM                                  *
      ****************************************************************
      *
       Z100-END.
      *
      *    RESET TRACE.

           CLOSE SYSTBLS
                 SYSTBLSP
                 SYSINDXS
                 SYSCOLMS
                 SYSCOLDT
                 REPOUT.
      *
       Z100-END-EXIT.
           EXIT.
```

PCATV3UP

```
 IDENTIFICATION DIVISION.
       PROGRAM-ID. PCATV3UP.
       AUTHOR. IOLANDA LOPES.
      *
      *   ___________________________________
      *   -   ****** PROGRAM THAT UPDATES THE COLUMNS OF   *****    -
      *   -   ****** A SPECIFIED DB2 CATALOG TABLES, ONLY  *****    -
      *   -   ****** THOSE THAT INFLUENCE THE ACCESS PATH  *****    -
      *   ___________________________________
      *
      *
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. IBM.
       OBJECT-COMPUTER. IBM.
       SPECIAL-NAMES.
           CONSOLE IS CONSOLA
           DECIMAL-POINT IS COMMA.
      *
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
      *
           SELECT  SYSTBLS  ASSIGN TO SYSTBLS
                    FILE STATUS IS  FILE-STATUS.

           SELECT  SYSTBLSP ASSIGN TO SYSTBLSP
                    FILE STATUS IS  FILE-STATUS.
```

```
     SELECT  SYSINDXS ASSIGN TO SYSINDXS
              FILE STATUS IS  FILE-STATUS.

     SELECT  SYSCOLMS ASSIGN TO SYSCOLMS
              FILE STATUS IS  FILE-STATUS.

     SELECT  SYSCOLDT ASSIGN TO SYSCOLDT
              FILE STATUS IS  FILE-STATUS.

     SELECT  REPOUT   ASSIGN TO REPOUT.
*
 DATA DIVISION.
 FILE SECTION.
*
*
 FD  SYSTBLS
     RECORD CONTAINS  Ø CHARACTERS
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS R-SYSTBLS.

 Ø1  R-SYSTBLS          PIC X(4Ø).
*
 FD  SYSTBLSP
     RECORD CONTAINS  Ø CHARACTERS
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS R-SYSTBLSP.

 Ø1  R-SYSTBLSP         PIC X(17).
*
*
 FD  SYSINDXS
     RECORD CONTAINS  Ø CHARACTERS
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS R-SYSINDXS.

 Ø1  R-SYSINDXS         PIC X(71).
*
 FD  SYSCOLMS
     RECORD CONTAINS  Ø CHARACTERS
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS R-SYSCOLMS.

 Ø1  R-SYSCOLMS         PIC X(69).
*
 FD  SYSCOLDT
     RECORD CONTAINS  Ø CHARACTERS
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS R-SYSCOLDT.

 Ø1  R-SYSCOLDT         PIC X(294).
```

```
*
 FD  REPOUT
     RECORD CONTAINS 12Ø CHARACTERS
     LABEL RECORDS ARE OMITTED
     DATA RECORD IS REPREC.
 Ø1  REPREC               PIC X(12Ø).
*
*
  WORKING-STORAGE SECTION.
*
*
 Ø1  FILE-STATUS        PIC 99.
*
 Ø1  ERROR-MESSAGE.
     Ø2  ERROR-LEN      PIC S9(4)  COMP VALUE +96Ø.
     Ø2  ERROR-TEXT     PIC X(12Ø) OCCURS 8 TIMES
                                   INDEXED BY ERROR-INDEX.
 77  ERROR-TEXT-LEN     PIC S9(4)  COMP VALUE +12Ø.
*
*———————————————————————————————
*    DB2 ERROR MESSAGE
*———————————————————————————————
*
 Ø1  MSG2.
     Ø2 NAME-TABLE            PIC X(5).
     Ø2 FILLER                PIC X(13) VALUE ' SQLCODE-ERR'.
     Ø2 ERROR-CODE            PIC -9(9).
*
*———————————————————————————————
*    WORKAREAS                                                  -
*———————————————————————————————
*
*
 Ø1  W-SYSTBLS.
     Ø5  W-TBL-NAME             PIC X(18).
     Ø5  W-TBL-CARD             PIC X(9).
     Ø5  W-TBL-NPAGES           PIC X(9).
     Ø5  W-TBL-PCTRCOMP         PIC X(4).
*
 Ø1  W-TBL-CARD-NUM             PIC 9(9).
 Ø1  W-TBL-CARD-NUM-R      REDEFINES
     W-TBL-CARD-NUM             PIC S9(9) COMP SYNC.
*
 Ø1  W-TBL-NPAGES-NUM           PIC 9(9).
 Ø1  W-TBL-NPAGES-NUM-R      REDEFINES
     W-TBL-NPAGES-NUM           PIC S9(9) COMP SYNC.
*
 Ø1  W-TBL-PCTRCOMP-NUM         PIC 9(4).
 Ø1  W-TBL-PCTRCOMP-NUM-R  REDEFINES
     W-TBL-PCTRCOMP-NUM         PIC S9(4) COMP SYNC.
```

```
*
 Ø1  W-SYSTBLSP.
     Ø5  W-TBLSP-NAME           PIC X(8).
     Ø5  W-TBLSP-NACTIVE        PIC X(9).
*
 Ø1  W-TBLSP-NACTIVE-NUM        PIC 9(9).
 Ø1  W-TBLSP-NACTIVE-NUM-R   REDEFINES
     W-TBLSP-NACTIVE-NUM        PIC S9(9) COMP SYNC.
*
 Ø1  W-SYSINDXS.
     Ø5  W-INDXS-TBNAME         PIC X(18).
     Ø5  W-INDXS-NAME           PIC X(18).
     Ø5  W-CLUSTERRATIO         PIC X(4).
     Ø5  W-FIRSTKEY-CARD        PIC X(9).
     Ø5  W-FULLKEY-CARD         PIC X(9).
     Ø5  W-NLEAF                PIC X(9).
     Ø5  W-NLEVELS              PIC X(4).
*
 Ø1  W-CLUSTERRATIO-NUM         PIC 9(4).
 Ø1  W-CLUSTERRAT-NUM-R      REDEFINES
     W-CLUSTERRATIO-NUM         PIC S9(4) COMP SYNC.
*
 Ø1  W-FIRSTKEY-CARD-NUM        PIC 9(9).
 Ø1  W-FIRSTKEY-NUM-R        REDEFINES
     W-FIRSTKEY-CARD-NUM        PIC S9(9) COMP SYNC.
*
 Ø1  W-FULLKEY-CARD-NUM         PIC 9(9).
 Ø1  W-FULLKEY-NUM-R         REDEFINES
     W-FULLKEY-CARD-NUM         PIC S9(9) COMP SYNC.
*
 Ø1  W-NLEAF-NUM                PIC 9(9).
 Ø1  W-NLEAF-NUM-R           REDEFINES
     W-NLEAF-NUM                PIC S9(9) COMP SYNC.
*
 Ø1  W-NLEVELS-NUM              PIC 9(4).
 Ø1  W-NLEVELS-NUM-R         REDEFINES
     W-NLEVELS-NUM              PIC S9(4) COMP SYNC.
*
  Ø1  W-SYSCOLMS.
     Ø5  W-COLMS-TBNAME         PIC X(18).
     Ø5  W-COLMS-NAME           PIC X(18).
     Ø5  W-COLCARD              PIC X(9).
     Ø5  W-LOW2KEY              PIC X(8).
     Ø5  W-HIGH2KEY             PIC X(8).
     Ø5  W-COLTYPE              PIC X(8).
*
 Ø1  W-COLCARD-NUM              PIC 9(9).
 Ø1  W-COLCARD-NUM-R         REDEFINES
     W-COLCARD-NUM              PIC S9(9) COMP SYNC.
*
```

```
*
 Ø1  W-FETCH-SYSCOLDT.
     Ø5  W-FETCH-COLDT-TBNAME   PIC X(18)  VALUE SPACES.
     Ø5  W-FETCH-COLDT-NAME     PIC X(18)  VALUE SPACES.
     Ø5  W-FETCH-COLVALUE       PIC X(254).
     Ø5  W-FETCH-FREQUENC       PIC S9(4)  COMP SYNC.
 Ø1  W-PREV-TBNAME              PIC X(18)  VALUE SPACES.
 Ø1  W-PREV-COLNAME             PIC X(18)  VALUE SPACES.
*
 Ø1  W-SYSCOLDT.
     Ø5  W-COLDT-TBNAME         PIC X(18).
     Ø5  W-COLDT-NAME           PIC X(18).
     Ø5  W-COLVALUE             PIC X(254).
     Ø5  W-FREQUENC             PIC X(4).
*
 Ø1  W-FREQUENC-NUM             PIC 9(4).
 Ø1  W-FREQUENC-NUM-R        REDEFINES
     W-FREQUENC-NUM             PIC S9(4) COMP SYNC.
*
*
 Ø1  WS-SWITCHES.
     Ø5  PROCESS-END            PIC 9(1) VALUE Ø.
*
*
 Ø1  COUNT-TBL-UPD              PIC 9(9) VALUE Ø.
*
*
*———————————————————————————————
*    INCLUDE SQL COMMUNICATION AREA                              -
*———————————————————————————————
*
     EXEC SQL
              INCLUDE SQLCA
     END-EXEC.
*
*
 Ø1  W-PARM.
     Ø5  W-PARM-DBNAME     PIC X(8).
     Ø5  W-PARM-TBOWNER    PIC X(8).
*
*
*
*
 PROCEDURE DIVISION.
*
*
     EXEC SQL
        WHENEVER SQLERROR GO TO X1ØØ-DBERROR-DB2
     END-EXEC.
*
```

```
*
     EXEC SQL
        WHENEVER SQLWARNING GO TO X100-DBERROR-DB2
     END-EXEC.
*
*
     EXEC SQL
        WHENEVER NOT FOUND CONTINUE
     END-EXEC.
*
*
     EXEC SQL DECLARE FETCH_SYSCOLDT CURSOR FOR
         SELECT TBNAME, NAME, COLVALUE, FREQUENCY
           FROM  SYSIBM.SYSCOLDIST
          WHERE TBOWNER = :W-PARM-TBOWNER
            AND TBNAME = :W-COLDT-TBNAME
            AND NAME = :W-COLDT-NAME
            FOR UPDATE OF COLVALUE, FREQUENCY
     END-EXEC.
*
```

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

Iolanda Lopes
Database Administrator
Companhia Seguros Mundial Confiança (Portugal)

Call for papers

Why not share your expertise and earn money at the same time? *DB2 Update* is looking for REXX EXECs, macros, program code, etc, that experienced DB2 users have written to make their life easier. We will publish them (after vetting by our expert panel) and send you a cheque when the article is published. Articles can be of any length and can be sent or e-mailed to Robert Burgess at any of the addresses shown on page 2. Why not call now for a free copy of our *Notes for contributors*?

Capturing DISPLAY BUFFERPOOL output – part 2

This month we conclude the article presenting a method for capturing the output from the DISPLAY BUFFERPOOL command in a set of DB2 tables for manipulation with SQL.

```
      J=INDEX(IN_REC,'=');
      CHANGED=SUBSTR(IN_REC,J+1,12);
      CHANGED=TRANSLATE(CHANGED,BLNKABET,ALPHABET);
      CHANGED=ZEROIT(CHANGED,12);
      TEMP_REC='';
      TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
      J=INDEX(TEMP_REC,'=');
      MCHANGED=SUBSTR(TEMP_REC,J+1,12);
      MCHANGED=TRANSLATE(MCHANGED,BLNKABET,ALPHABET);
      MCHANGED=ZEROIT(MCHANGED,12);
      OUT_REC='';
      WRITE FILE(OUT) FROM(OUT_453);
   ENDDO;
   ENDIF;
   I=INDEX(IN_REC,'DSNB455I');
   IF I>Ø THEN DO;
      OUT_45N='';
      TYPE45N='Ø3';
      BP45N=BPNAME;
      OUT_45X_N=NAME45X||FILL45X_1||USECOUNT||FILL45X_2||PARTNUM;
      RTIME45N=DATETIME;
      IN_REC='';
     READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB455 MESSAGE */
      J=INDEX(IN_REC,'=');
      AVGDELAY=SUBSTR(IN_REC,J+1,12);
      AVGDELAY=TRANSLATE(AVGDELAY,BLNKABET,ALPHABET);
      AVGDELAY=ZEROIT(AVGDELAY,12);
      TEMP_REC='';
      TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
      J=INDEX(TEMP_REC,'=');
      MAXDELAY=SUBSTR(TEMP_REC,J+1,12);
      MAXDELAY=TRANSLATE(MAXDELAY,BLNKABET,ALPHABET);
      MAXDELAY=ZEROIT(MAXDELAY,12);
      IN_REC='';
     READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB455 MESSAGE */
      J=INDEX(IN_REC,'=');
      TOTPGS=SUBSTR(IN_REC,J+1,12);
      TOTPGS=TRANSLATE(TOTPGS,BLNKABET,ALPHABET);
      TOTPGS=ZEROIT(TOTPGS,12);
      OUT_REC='';
      WRITE FILE(OUT) FROM(OUT_45N);
```

```
ENDDO;
ENDIF;
I=INDEX(IN_REC,'DSNB456I');
IF I>Ø THEN DO;
    OUT_45N='';
    TYPE45N='Ø4';
    BP45N=BPNAME;
    OUT_45X_N=NAME45X||FILL45X_1||USECOUNT||FILL45X_2||PARTNUM;
    RTIME45N=DATETIME;
    IN_REC='';
  READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB456 MESSAGE */
    J=INDEX(IN_REC,'=');
    AVGDELAY=SUBSTR(IN_REC,J+1,12);
    AVGDELAY=TRANSLATE(AVGDELAY,BLNKABET,ALPHABET);
    AVGDELAY=ZEROIT(AVGDELAY,12);
    TEMP_REC='';
    TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
    J=INDEX(TEMP_REC,'=');
    MAXDELAY=SUBSTR(TEMP_REC,J+1,12);
    MAXDELAY=TRANSLATE(MAXDELAY,BLNKABET,ALPHABET);
    MAXDELAY=ZEROIT(MAXDELAY,12);
    IN_REC='';
  READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB456 MESSAGE */
    J=INDEX(IN_REC,'=');
    TOTPGS=SUBSTR(IN_REC,J+1,12);
    TOTPGS=TRANSLATE(TOTPGS,BLNKABET,ALPHABET);
    TOTPGS=ZEROIT(TOTPGS,12);
    TEMP_REC='';
    TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
    J=INDEX(TEMP_REC,'=');
    TOTIOS=SUBSTR(TEMP_REC,J+1,12);
    TOTIOS=TRANSLATE(TOTIOS,BLNKABET,ALPHABET);
    TOTIOS=ZEROIT(TOTIOS,12);
    OUT_REC='';
    WRITE FILE(OUT) FROM(OUT_45N);
ENDDO;
ENDIF;
I=INDEX(IN_REC,'DSNB41ØI');
IF I>Ø THEN DO;
    OUT_41Ø='';
    J=INDEX(IN_REC,'SINCE');
    CUMTIME=SUBSTR(IN_REC,J+6,25);
    IN_REC='';
  READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB411 MESSAGE */
    OUT_41X='';
    TYPE41X='Ø5';
    BP41X=BPNAME;
    RTIME41X=DATETIME;
    TIME41X=CUMTIME;
    J=INDEX(IN_REC,'=');
```

```
  RGETPAGE=SUBSTR(IN_REC,J+1,12);
  RGETPAGE=TRANSLATE(RGETPAGE,BLNKABET,ALPHABET);
  RGETPAGE=ZEROIT(RGETPAGE,12);
  TEMP_REC='';
  TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
  J=INDEX(TEMP_REC,'=');
  RSYNCIO=SUBSTR(TEMP_REC,J+1,12);
  RSYNCIO=TRANSLATE(RSYNCIO,BLNKABET,ALPHABET);
  RSYNCIO=ZEROIT(RSYNCIO,12);
  IN_REC='';
READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB411 MESSAGE */
  J=INDEX(IN_REC,'=');
  SGETPAGE=SUBSTR(IN_REC,J+1,12);
  SGETPAGE=TRANSLATE(SGETPAGE,BLNKABET,ALPHABET);
  SGETPAGE=ZEROIT(SGETPAGE,12);
  TEMP_REC='';
  TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
  J=INDEX(TEMP_REC,'=');
  SSYNCIO=SUBSTR(TEMP_REC,J+1,12);
  SSYNCIO=TRANSLATE(SSYNCIO,BLNKABET,ALPHABET);
  SSYNCIO=ZEROIT(SSYNCIO,12);
  IN_REC='';
READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB411 MESSAGE */
  J=INDEX(IN_REC,'=');
  DMTHHIT=SUBSTR(IN_REC,J+1,12);
  DMTHHIT=TRANSLATE(DMTHHIT,BLNKABET,ALPHABET);
  DMTHHIT=ZEROIT(DMTHHIT,12);
IN_REC='';
READ FILE(IN) INTO(IN_REC); /* PROCESS DSNB412 MESSAGE */
IN_REC='';
READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB412 MESSAGE */
  J=INDEX(IN_REC,'=');
  SPREQST=SUBSTR(IN_REC,J+1,12);
  SPREQST=TRANSLATE(SPREQST,BLNKABET,ALPHABET);
  SPREQST=ZEROIT(SPREQST,12);
  TEMP_REC='';
  TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
  J=INDEX(TEMP_REC,'=');
  SPIOS=SUBSTR(TEMP_REC,J+1,12);
  SPIOS=TRANSLATE(SPIOS,BLNKABET,ALPHABET);
  SPIOS=ZEROIT(SPIOS,12);
IN_REC='';
READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB412 MESSAGE */
  J=INDEX(IN_REC,'=');
  SPPGREAD=SUBSTR(IN_REC,J+1,12);
  SPPGREAD=TRANSLATE(SPPGREAD,BLNKABET,ALPHABET);
  SPPGREAD=ZEROIT(SPPGREAD,12);
IN_REC='';
READ FILE(IN) INTO(IN_REC); /* PROCESS DSNB413 MESSAGE */
IN_REC='';
```

```
READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB413 MESSAGE */
  J=INDEX(IN_REC,'=');
  LPREQST=SUBSTR(IN_REC,J+1,12);
  LPREQST=TRANSLATE(LPREQST,BLNKABET,ALPHABET);
  LPREQST=ZEROIT(LPREQST,12);
  TEMP_REC='';
  TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
  J=INDEX(TEMP_REC,'=');
  LPIOS=SUBSTR(TEMP_REC,J+1,12);
  LPIOS=TRANSLATE(LPIOS,BLNKABET,ALPHABET);
  LPIOS=ZEROIT(LPIOS,12);
IN_REC='';
READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB413 MESSAGE */
  J=INDEX(IN_REC,'=');
  LPPGREAD=SUBSTR(IN_REC,J+1,12);
  LPPGREAD=TRANSLATE(LPPGREAD,BLNKABET,ALPHABET);
  LPPGREAD=ZEROIT(LPPGREAD,12);
IN_REC='';
READ FILE(IN) INTO(IN_REC); /* PROCESS DSNB414 MESSAGE */
IN_REC='';
READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB414 MESSAGE */
  J=INDEX(IN_REC,'=');
  DPREQST=SUBSTR(IN_REC,J+1,12);
  DPREQST=TRANSLATE(DPREQST,BLNKABET,ALPHABET);
  DPREQST=ZEROIT(DPREQST,12);
  TEMP_REC='';
  TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
  J=INDEX(TEMP_REC,'=');
  DPIOS=SUBSTR(TEMP_REC,J+1,12);
  DPIOS=TRANSLATE(DPIOS,BLNKABET,ALPHABET);
  DPIOS=ZEROIT(DPIOS,12);
IN_REC='';
READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB414 MESSAGE */
  J=INDEX(IN_REC,'=');
  DPPGREAD=SUBSTR(IN_REC,J+1,12);
  DPPGREAD=TRANSLATE(DPPGREAD,BLNKABET,ALPHABET);
  DPPGREAD=ZEROIT(DPPGREAD,12);
IN_REC='';
READ FILE(IN) INTO(IN_REC); /* PROCESS DSNB415 MESSAGE */
IN_REC='';
READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB415 MESSAGE */
  J=INDEX(IN_REC,'=');
  PFNOBUF=SUBSTR(IN_REC,J+1,12);
  PFNOBUF=TRANSLATE(PFNOBUF,BLNKABET,ALPHABET);
  PFNOBUF=ZEROIT(PFNOBUF,12);
  TEMP_REC='';
  TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
  J=INDEX(TEMP_REC,'=');
  PFNORDE=SUBSTR(TEMP_REC,J+1,12);
  PFNORDE=TRANSLATE(PFNORDE,BLNKABET,ALPHABET);
```

```
    PFNORDE=ZEROIT(PFNORDE,12);
    OUT_REC='';
    WRITE FILE(OUT) FROM(OUT_41X);
ENDDO;
ENDIF;
I=INDEX(IN_REC,'DSNB42ØI'); /* PROCESS DSNB42Ø MESSAGE */
IF I>Ø THEN DO;
    OUT_42X='';
    TYPE42X='Ø6';
    RTIME42X=DATETIME;
    BP42X=BPNAME;
    TIME42X=CUMTIME;
    J=INDEX(IN_REC,'=');
    SYSPGUPD=SUBSTR(IN_REC,J+1,12);
    SYSPGUPD=TRANSLATE(SYSPGUPD,BLNKABET,ALPHABET);
    SYSPGUPD=ZEROIT(SYSPGUPD,12);
    TEMP_REC='';
    TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
    J=INDEX(TEMP_REC,'=');
    SYSPGWR=SUBSTR(TEMP_REC,J+1,12);
    SYSPGWR=TRANSLATE(SYSPGWR,BLNKABET,ALPHABET);
    SYSPGWR=ZEROIT(SYSPGWR,12);
  IN_REC='';
  READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB42Ø MESSAGE */
    J=INDEX(IN_REC,'=');
    ASYNCWIO=SUBSTR(IN_REC,J+1,12);
    ASYNCWIO=TRANSLATE(ASYNCWIO,BLNKABET,ALPHABET);
    ASYNCWIO=ZEROIT(ASYNCWIO,12);
    TEMP_REC='';
    TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
    J=INDEX(TEMP_REC,'=');
    SYNCWIO=SUBSTR(TEMP_REC,J+1,12);
    SYNCWIO=TRANSLATE(SYNCWIO,BLNKABET,ALPHABET);
    SYNCWIO=ZEROIT(SYNCWIO,12);
  IN_REC='';
  READ FILE(IN) INTO(IN_REC); /* PROCESS DSNB421 MESSAGE */
    J=INDEX(IN_REC,'=');
    DWTHIT=SUBSTR(IN_REC,J+1,12);
    DWTHIT=TRANSLATE(DWTHIT,BLNKABET,ALPHABET);
    DWTHIT=ZEROIT(DWTHIT,12);
    TEMP_REC='';
    TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
    J=INDEX(TEMP_REC,'=');
    VDWTHIT=SUBSTR(TEMP_REC,J+1,12);
    VDWTHIT=TRANSLATE(VDWTHIT,BLNKABET,ALPHABET);
    VDWTHIT=ZEROIT(VDWTHIT,12);
  IN_REC='';
  READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB421 MESSAGE */
    J=INDEX(IN_REC,'=');
    NOWRENG=SUBSTR(IN_REC,J+1,12);
```

```
    NOWRENG=TRANSLATE(NOWRENG,BLNKABET,ALPHABET);
    NOWRENG=ZEROIT(NOWRENG,12);
    OUT_REC='';
    WRITE FILE(OUT) FROM(OUT_42X);
ENDDO;
ENDIF;
I=INDEX(IN_REC,'DSNB430I'); /* PROCESS DSNB430 MESSAGE */
IF I>0 THEN DO;
  READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB430 MESSAGE */
  IN_REC='';
  READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB430 MESSAGE */
    OUT_43Y='';
    TYPE43Y='07';
    BP43Y=BPNAME;
    RTIME43Y=DATETIME;
    TIME43Y=CUMTIME;
    J=INDEX(IN_REC,'=');
    NSYNCRD=SUBSTR(IN_REC,J+1,12);
    NSYNCRD=TRANSLATE(NSYNCRD,BLNKABET,ALPHABET);
    NSYNCRD=ZEROIT(NSYNCRD,12);
    TEMP_REC='';
    TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
    J=INDEX(TEMP_REC,'=');
    NSYNCWR=SUBSTR(TEMP_REC,J+1,12);
    NSYNCWR=TRANSLATE(NSYNCWR,BLNKABET,ALPHABET);
    NSYNCWR=ZEROIT(NSYNCWR,12);
  IN_REC='';
  READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB430 MESSAGE */
    J=INDEX(IN_REC,'=');
    NASYNCRD=SUBSTR(IN_REC,J+1,12);
    NASYNCRD=TRANSLATE(NASYNCRD,BLNKABET,ALPHABET);
    NASYNCRD=ZEROIT(NASYNCRD,12);
    TEMP_REC='';
    TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
    J=INDEX(TEMP_REC,'=');
    NASYNCWR=SUBSTR(TEMP_REC,J+1,12);
    NASYNCWR=TRANSLATE(NASYNCWR,BLNKABET,ALPHABET);
    NASYNCWR=ZEROIT(NASYNCWR,12);
  IN_REC='';
  READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB430 MESSAGE */
    J=INDEX(IN_REC,'=');
    NRDFAIL=SUBSTR(IN_REC,J+1,12);
    NRDFAIL=TRANSLATE(NRDFAIL,BLNKABET,ALPHABET);
    NRDFAIL=ZEROIT(NRDFAIL,12);
    TEMP_REC='';
    TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
    J=INDEX(TEMP_REC,'=');
    NWRFAIL=SUBSTR(TEMP_REC,J+1,12);
    NWRFAIL=TRANSLATE(NWRFAIL,BLNKABET,ALPHABET);
    NWRFAIL=ZEROIT(NWRFAIL,12);
```

```
      READ FILE(IN) INTO(IN_REC); /* PROCESS DSNB431 MESSAGE */
      READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB431 MESSAGE */
      IN_REC='';
      READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB431 MESSAGE */
        J=INDEX(IN_REC,'=');
        UREADS=SUBSTR(IN_REC,J+1,12);
        UREADS=TRANSLATE(UREADS,BLNKABET,ALPHABET);
        UREADS=ZEROIT(UREADS,12);
        TEMP_REC='';
        TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
        J=INDEX(TEMP_REC,'=');
        UWRITES=SUBSTR(TEMP_REC,J+1,12);
        UWRITES=TRANSLATE(UWRITES,BLNKABET,ALPHABET);
        UWRITES=ZEROIT(UWRITES,12);
      IN_REC='';
      READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB431 MESSAGE */
        J=INDEX(IN_REC,'=');
        URDFAIL=SUBSTR(IN_REC,J+1,12);
        URDFAIL=TRANSLATE(URDFAIL,BLNKABET,ALPHABET);
        URDFAIL=ZEROIT(URDFAIL,12);
        TEMP_REC='';
        TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
        J=INDEX(TEMP_REC,'=');
        UWRFAIL=SUBSTR(TEMP_REC,J+1,12);
        UWRFAIL=TRANSLATE(UWRFAIL,BLNKABET,ALPHABET);
        UWRFAIL=ZEROIT(UWRFAIL,12);
      READ FILE(IN) INTO(IN_REC); /* PROCESS DSNB44Ø MESSAGE */
      IN_REC='';
      READ FILE(IN) INTO(IN_REC); /* CONTINUE WITH DSNB44Ø MESSAGE */
        J=INDEX(IN_REC,'=');
        IOPRQST=SUBSTR(IN_REC,J+1,12);
        IOPRQST=TRANSLATE(IOPRQST,BLNKABET,ALPHABET);
        IOPRQST=ZEROIT(IOPRQST,12);
        TEMP_REC='';
        TEMP_REC=SUBSTR(IN_REC,J+1,137-J);
        J=INDEX(TEMP_REC,'=');
        IOPDEGRAD=SUBSTR(TEMP_REC,J+1,12);
        IOPDEGRAD=TRANSLATE(IOPDEGRAD,BLNKABET,ALPHABET);
        IOPDEGRAD=ZEROIT(IOPDEGRAD,12);
        OUT_REC='';
        WRITE FILE(OUT) FROM(OUT_43Y);
    ENDDO;
    ENDIF;
    IN_REC='';
    READ FILE(IN) INTO(IN_REC);
  ENDDO;
  CLOSE FILE(IN);
  CLOSE FILE(OUT);
  ZEROIT: PROC (INPUTREC,LENGTH) RETURNS (CHAR(12) VARYING);
    DCL INPUTREC       CHAR(12);
```

```
   DCL LENGTH         FIXED BIN(15,Ø);
   DCL (K,L)          FIXED BIN(15,Ø) INIT(Ø);
   DCL DIGITS         CHAR(9)         INIT('123456789');
   DCL BLANK          CHAR(1)         INIT(' ');
   DCL ZERO           CHAR(1)         INIT('Ø');
   DCL STINGOUT       CHAR(12) VARYING INIT('            ');
   K=LENGTH;
   L=LENGTH;
   DO WHILE (K>Ø) ;
     IF SUBSTR(INPUTREC,K,1)¬=' ' THEN
     DO;
       SUBSTR(STINGOUT,L,1)=SUBSTR(INPUTREC,K,1);
       K=K-1;
       L=L-1;
     ENDIF;
     ELSE
     DO;
       K=K-1;
     ENDDO;
     ENDDO;
   ENDDO;
   STINGOUT=TRANSLATE(STINGOUT,ZERO,BLANK);
   RETURN(SUBSTR(STINGOUT,1,LENGTH));
 END ZEROIT;
 END PALØP;
```

IKJEFT01

```
//DISPLY  EXEC PGM=IKJEFTØ1
//STEPLIB  DD   DISP=SHR,DSN=DB2 load library
//SYSTSPRT DD DISP=(,CATLG),DSN=output1,
//         SPACE=(TRK,(5,5),RLSE),
//         DCB=(RECFM=VBA,LRECL=137,BLKSIZE=27998)
//SYSPRINT DD   SYSOUT=*
//SYSTSIN  DD   *
DSN SYSTEM(DSN)
-DIS BUFFERPOOL(ACTIVE) DETAIL(INTERVAL) LSTATS
END
//*
//PALØP    EXEC PGM=PALØP
//STEPLIB  DD DISP=SHR,DSN=user load library
//IN       DD DISP=SHR,DSN=output1
//OUT      DD DSN=output2,
//         DISP=(NEW,CATLG,CATLG),SPACE=(TRK,(5,5),RLSE),
//         DCB=(RECFM=FB,LRECL=3ØØ,BLKSIZE=3ØØØØ)
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//PLIDUMP  DD SYSOUT=*
```

EXAMPLE TO JOIN VIEWS

```
/*                                                          */ ;
/*  MACRO TO REPLACE SELECTION CRITERIA ————————*/ ;
/*                                                          */ ;
  %MACRO OBNMSEL;
   WHERE A.XXXXOBNM NOT LIKE 'DSN%'
     AND A.XXXXOBNM NOT LIKE 'APS%'
     AND A.XXXXOBNM NOT LIKE 'IE%'
     AND A.XXXXSTME LIKE '97Ø2%'
  %MEND OBNMSEL;
/*                                                          */ ;
/*  BUILD ACCESS AND VIEW DESCRIPTORS ——————————*/ ;
/*                                                          */ ;
PROC ACCESS DBMS=DB2 ;
   CREATE WORK.DSNB45XI.ACCESS ;
   TABLE=TØ8DSNB45XI ;
   SSID=DSN ;
   ASSIGN=YES ;
   LIST ALL ;
   CREATE WORK.DSNB45XI.VIEW ;
   SELECT ALL ;
   LIST ALL ;
   CREATE WORK.DSNB455I.ACCESS ;
   TABLE=TØ9DSNB455I ;
   SSID=DSN ;
   ASSIGN=YES ;
   LIST ALL ;
   CREATE WORK.DSNB455I.VIEW ;
   SELECT ALL ;
   LIST ALL ;
   CREATE WORK.DSNB456I.ACCESS ;
   TABLE=T1ØDSNB456I ;
   SSID=DSN ;
   ASSIGN=YES ;
   LIST ALL ;
   CREATE WORK.DSNB456I.VIEW ;
   SELECT ALL ;
   LIST ALL ;
RUN ;
/*                                                          */ ;
/*  EXEC SQL, JOIN/UNION WITH THE VIEWS ABOVE ———*/ ;
/*                                                          */ ;
PROC SQL ;
 CREATE TABLE WORK.BPØ1 AS
 SELECT A.XXXXOBNM AS OBJECT,
        SUBSTR(A.XXXXSTME,1,6) AS RPTDATE,
        SUBSTR(A.XXXXSTME,7,4) AS RPTTIME,
        A.XXXXOBPN AS OBJPART, A.XXXXTPGS AS SYNCRWPG,
        C.XXXXCCAC AS CURCACHE, C.XXXXCHNG AS CURCHNG,
```

```
        C.XXXXOBUC AS OBJCOUNT,
        C.XXXXMCAC AS MAXCACHE, C.XXXXMCHG AS MAXCHNG,
        Ø  AS ASYNRWPG, Ø  AS ASYNIOS
   FROM DSNB455I A,
        DSNB45XI C
   %OBNMSEL
    AND A.XXXXSTME = C.XXXXSTME
    AND A.XXXXOBNM = C.XXXXOBNM
    AND A.XXXXOBPN = C.XXXXOBPN
    AND NOT EXISTS ( SELECT 1
   FROM DSNB456I B
  WHERE A.XXXXSTME = B.XXXXSTME
    AND A.XXXXOBNM = B.XXXXOBNM
    AND A.XXXXOBPN = B.XXXXOBPN )
 UNION
 SELECT A.XXXXOBNM AS OBJECT,
        SUBSTR(A.XXXXSTME,1,6) AS RPTDATE,
        SUBSTR(A.XXXXSTME,7,4) AS RPTTIME,
        A.XXXXOBPN AS OBJPART, A.XXXXTPGS AS SYNCRWPG,
        C.XXXXCCAC AS CURCACHE, C.XXXXCHNG AS CURCHNG,
        C.XXXXOBUC AS OBJCOUNT,
        C.XXXXMCAC AS MAXCACHE, C.XXXXMCHG AS MAXCHNG,
        B.XXXXTPGS AS ASYNRWPG, B.XXXXTIOS AS ASYNIOS
   FROM DSNB455I A,
        DSNB456I B,
        DSNB45XI C
   %OBNMSEL
    AND A.XXXXSTME = B.XXXXSTME
    AND A.XXXXOBNM = B.XXXXOBNM
    AND A.XXXXOBPN = B.XXXXOBPN
    AND A.XXXXSTME = C.XXXXSTME
    AND A.XXXXOBNM = C.XXXXOBNM
    AND A.XXXXOBPN = C.XXXXOBPN
  ORDER BY 2 ,3  ;
/*                                                          */ ;
/*  PRINT THE REPORT ——————————————————*/ ;
/*                                                          */ ;
PROC PRINT DATA=WORK.BPØ1 NOOBS ;
 VAR RPTDATE RPTTIME OBJECT OBJPART
     SYNCRWPG CURCACHE MAXCACHE CURCHNG MAXCHNG ASYNRWPG ASYNIOS ;
 SUM SYNCRWPG CURCACHE MAXCACHE CURCHNG MAXCHNG ASYNRWPG ASYNIOS ;
RUN ;
/*                                                          */ ;
/*  USE PROC MEANS TO GET _FREQ_ AND SUM VALUES ——*/ ;
/*                                                          */ ;
PROC MEANS DATA=WORK.BPØ1 NOPRINT ;
 CLASS RPTDATE RPTTIME ;
 VAR OBJCOUNT
     SYNCRWPG CURCACHE MAXCACHE CURCHNG MAXCHNG ASYNRWPG ASYNIOS ;
 OUTPUT OUT=WORK.BPØ11
```

```
 SUM = OUCNT SRWPG CCACHE MCACHE CCHNG MCHNG ARWPG AIOS ;
RUN ;
/*                                                          */ ;
/*  THROW OUT THE THINGS NOT NEEDED ————————*/ ;
/*                                                          */ ;
DATA WORK.BPØ12
     (KEEP= RPTDATE RPTTIME
            OUCNT SRWPG CCACHE MCACHE CCHNG MCHNG ARWPG AIOS) ;
 IF _TYPE_ = 3 THEN OUTPUT ;
 SET WORK.BPØ11 ;
/*                                                          */ ;
/*  NOW SORT THE DATA ——————————————————*/ ;
/*                                                          */ ;
PROC SORT DATA=WORK.BPØ12 OUT=WORK.BPØ13 ;
 BY RPTDATE RPTTIME ;
RUN ;
/*                                                          */ ;
/*  NOW PRINT THE STUFF ———————————————*/ ;
/*                                                          */ ;
PROC PRINT DATA=WORK.BPØ13 NOOBS ;
 VAR  RPTDATE RPTTIME
      OUCNT SRWPG CCACHE MCACHE CCHNG MCHNG ARWPG AIOS ;
RUN ;
XX
```

EXAMPLE OF VIRTUAL BUFFERPOOL RATIOS

```
/*                                                          */ ;
/*  MACRO TO REPLACE SELECTION CRITERIA ————————*/ ;
/*                                                          */ ;
  %MACRO OBNMSEL;
   WHERE A.XXXXSTME LIKE '97Ø2%'
     AND A.XXXXBPNM = 'BPØ '
  %MEND OBNMSEL;
/*                                                          */ ;
/*  BUILD ACCESS AND VIEW DESCRIPTORS ————————*/ ;
/*                                                          */ ;
PROC ACCESS DBMS=DB2 ;
   CREATE WORK.DSNB41XI.ACCESS ;
   TABLE=TØ6DSNB41XI ;
   SSID=DSN ;
   ASSIGN=YES ;
   LIST ALL ;
   CREATE WORK.DSNB41XI.VIEW ;
   SELECT ALL ;
   LIST ALL ;
   CREATE WORK.DSNB42XI.ACCESS ;
   TABLE=TØ4DSNB42XI ;
   SSID=DSN ;
```

```
   ASSIGN=YES ;
   LIST ALL ;
   CREATE WORK.DSNB42XI.VIEW ;
   SELECT ALL ;
   LIST ALL ;
RUN ;
/*                                                          */ ;
/*  EXEC SQL, SELECT FROM VIEW DSNB41XI ————————*/ ;
/*                                                          */ ;
PROC SQL ;
 CREATE TABLE WORK.BPØ1 AS
 SELECT SUBSTR(A.XXXXSTME,7,4) AS RPTTIME,
        SUBSTR(A.XXXXSTME,1,6) AS RPTDATE,
        A.XXXXRGPG, A.XXXXRSIO,
        A.XXXXSGPG, A.XXXXSSIO
   FROM DSNB41XI A
   %OBNMSEL
  ORDER BY 1 ;
PROC SQL ;
 CREATE TABLE WORK.BPØ12 AS
 SELECT SUBSTR(A.XXXXSTME,7,4) AS RPTTIME,
        SUBSTR(A.XXXXSTME,1,6) AS RPTDATE,
        A.XXXXSYWI, A.XXXXDWTH,
        A.XXXXVDWT
   FROM DSNB42XI A
   %OBNMSEL
  ORDER BY 1 ;
/*                                                          */ ;
/*  SORT AND MERGE FOR STANDARD REPORT —————————*/ ;
/*                                                          */ ;
PROC SORT DATA=WORK.BPØ1 ;
  BY RPTDATE RPTTIME ;
PROC SORT DATA=WORK.BPØ12 ;
  BY RPTDATE RPTTIME ;
RUN ;
DATA WORK.BPØ1299 ;
 MERGE WORK.BPØ1 WORK.BPØ12 ;
 BY RPTDATE RPTTIME ;
/*                                                          */ ;
/*  PRINT THE REPORT ———————————————————*/ ;
/*                                                          */ ;
PROC PRINT DATA=WORK.BPØ1299 NOOBS ;
 VAR RPTDATE RPTTIME XXXXRGPG XXXXRSIO
     XXXXSGPG XXXXSSIO XXXXSYWI XXXXDWTH XXXXVDWT ;
RUN ;
/*                                                          */ ;
/*  CALCULATE THE DIFF VALUES ———————————————*/ ;
/*                                                          */ ;
DATA WORK.BPØ22
   (KEEP=SYWI VDWT DWTH RPTTIME RPTDATE) ;
```

```
 SET WORK.BPØ12 ;
 OSYWI  = LAG1(XXXXSYWI) ;
 SYWI   = DIF1(XXXXSYWI) ;
 ODWTH  = LAG1(XXXXDWTH) ;
 DWTH   = DIF1(XXXXDWTH) ;
 OVDWT  = LAG1(XXXXVDWT) ;
 VDWT   = DIF1(XXXXVDWT) ;
RUN ;
DATA WORK.BPØ2
   (KEEP=RGPG RSIO SGPG SSIO RPTTIME RPTDATE) ;
 SET WORK.BPØ1 ;
 ORGPG  = LAG1(XXXXRGPG) ;
 RGPG   = DIF1(XXXXRGPG) ;
 ORSIO  = LAG1(XXXXRSIO) ;
 RSIO   = DIF1(XXXXRSIO) ;
 OSGPG  = LAG1(XXXXSGPG) ;
 SGPG   = DIF1(XXXXSGPG) ;
 OSSIO  = LAG1(XXXXSSIO) ;
 SSIO   = DIF1(XXXXSSIO) ;
RUN ;
/*                                                          */ ;
/*  CALCULATE RATIOS ——————————————————*/ ;
/*                                                          */ ;
DATA WORK.BPØ3
   (KEEP=RPTDATE RGPG RSIO RHIT SGPG SSIO SHIT RPTTIME) ;
 SET WORK.BPØ2 ;
 RHIT   = 1ØØ*(RGPG-RSIO)/RGPG ;
 SHIT   = 1ØØ*(SGPG-SSIO)/SGPG ;
RUN ;
/*                                                          */ ;
/*  SORT AND MERGE ———————————————————*/ ;
/*                                                          */ ;
PROC SORT DATA=WORK.BPØ3 ;
  BY RPTDATE RPTTIME ;
PROC SORT DATA=WORK.BPØ22 ;
  BY RPTDATE RPTTIME ;
RUN ;
DATA WORK.BP999 ;
 MERGE WORK.BPØ22 WORK.BPØ3 ;
 BY RPTDATE RPTTIME ;
/*                                                          */ ;
/*  PRINT THE REPORT ——————————————————*/ ;
/*                                                          */ ;
PROC PRINT DATA=WORK.BP999 NOOBS ;
 VAR RPTDATE RPTTIME RGPG RSIO RHIT SGPG SSIO SHIT
     SYWI VDWT DWTH ;
RUN ;
XX
```

Informatica has announced native support for DB2 at the source, repository, and target levels on both Unix and Windows NT. PowerCenter 1.5 and PowerMart 4.5 – Informatica's software product for building, deploying, and managing scalable data marts and data warehouses – will allow users to deploy data marts and data warehouses in a heterogeneous environment with DB2 as the target database. In addition, DB2 can be designated as the platform for a user's metadata repository.

PowerCenter 1.5 includes a high-throughput mechanism for natively extracting data from DB2 on mainframes. An engine-based pump resident on the mainframe automatically moves data into target stores on Unix or Windows NT platforms.

For further information contact:
Informatica, 1200 Chrysler Drive, Menlo Park, CA 94025, USA.
Tel: (650) 462 8900.
Informatica, Chancery Court, Lincolns Inn, High Wycombe, Bucks, HP12 3RE, UK.
Tel: (0990) 275223.
URL: http://www.informatica.com.

* * *

Cognos has included new support for DB2 in Version 6.0 of its PowerPlay OLAP tools. Version 6.0 includes better reporting features, an ability to handle larger data sources, shorter build times for cubes, and deployment improvements such as multiple currency conversion and calculated categories that can be stored in the multidimensional PowerCube.

Among the deployment enhancements are a multilingual server in the Web edition, better report-authoring features in the Client version, and calculated categories for storing calculations used in reports directly in the cube, so they can be shared with thin clients. There are also currency rate conversion tables. Support for DB2 means PowerCubes can be stored inside databases that are compatible with DB2 Common Server 2.1 and DB2 Universal Database 5.

For further information contact:
Cognos, 67 South Bedford Street, Burlington, MA 01803-5164, USA.
Tel: (781) 229 6000.
Cognos, Westerly Point, Market Street, Bracknell, Berks, RG12 1QB, UK.
Tel: (01344) 486668.
URL: http://www.cognos.com.

* * *

IBM has announced DB2 Row Archive Manager for OS/390 for selecting, managing, archiving, and retrieving aged data in a DB2 environment. By applying appropriate judgement on aged data, DB2 Row Archive Manager can help save DASD storage, improve DB2 performance, and simplify database maintenance.

For further information contact your local IBM representative.

* * *