96

DB2 update

October 2000

In this issue

DB2 Update

Published by

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

North American office

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

Subscriptions and back-issues

A year's subscription to *DB2 Update*, comprising twelve monthly issues, costs £255.00 in the UK; $380.00 in the USA and Canada; £261.00 in Europe; £267.00 in Australasia and Japan; and £265.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1997 issue, are available separately to subscribers for £22.50 ($33.50) each including postage.

DB2 Update on-line

Code from *DB2 Update* can be downloaded from our Web site at http://www.xephon.com/db2update.html; you will need the user-id shown on your address label.

Editor

Trevor Eddolls

Disclaimer

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

Contributions

Articles published in *DB2 Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. 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*, or you can download a copy from www.xephon.com/contnote.html.

Printed in England.

Letting SQL do the work

INTRODUCTION

When a DBA is considering how to build input to DB2 utilities – such as RUNSTATS or REORG – where does he start? The DB2 catalog tables, of course: he needs to consider which objects to include in a given job, what options to use, etc.

Many articles have appeared in *DB2 Update* over the years on tools that aid this process, and most sites have implemented some form of dialog or script with this aim.

These often take the form of ISPF dialogs or REXX EXECs, which access the raw data in the DB2 catalog tables with an SQL SELECT and reformat it into the relevant utility input.

However, there is something of a short cut that can sometimes be exploited, particularly with the character functions that have been introduced into DB2 in recent releases – you can build the utility input with SQL alone, and run it with DSNTIAUL. This is likely to be significantly faster than using REXX or another interpretive language. In some cases this may not be important, but in others (such as when a DBA is having to manually control work as part of maintenance to be carried out within a fixed window, no doubt at overtime rates), it can be important. At one site, I have seen utility jobs built with REXX EXECs spend 30% of their elapsed time in REXX steps, building the input to the DSNUTIL step, which is actually carrying out the real work.

This approach is frequently seen in other DBMS environments – Oracle, for example, ship utility SQL scripts of this nature with their product in Unix and Windows NT environments. It offers good performance without requiring tailored utility decks to be maintained every time objects are changed within a database.

The following job is an example of this that I have used at a couple of sites: the SQL in the DSNTIAUL step generates RUNSTATS for different databases within a subsystem dependent on the date on which the job is run. Thus only one job need be scheduled to run (weekly, in this example), but allowing only part of the subsystem to be processed each time. If the job were likely to run around midnight, it might be worth considering modifying the use of CURRENT TIMESTAMP to, say, CURRENT TIMESTAMP – 5 HOURS: this would avoid potential problems of sequencing if the job ran either side of midnight.

```
//xxxxxx JOB . . .
//         REGION=6M
//*
//JOBLIB    DD DSN=DSN.SDSNLOAD,DISP=SHR
//*
//*
//*
//*    BUILD RUNSTATS INPUT DECK
//*
//STEP0010 EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(subsystem)
 RUN  PROGRAM(DSNTIAUL) PLAN(DSNTIAUL) -
      LIB('db2.RUNLIB.LOAD') PARM('SQL')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSREC00 DD DSN=&&REC00,DISP=(NEW,PASS),
//            SPACE=(CYL,(4,1),RLSE),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=0),
//            UNIT=WORK
//SYSPUNCH DD SYSOUT=*
//SYSIN    DD  *
  SELECT SUBSTR(
                'RUNSTATS TABLESPACE '||
                 STRIP(DBNAME,T,' ')||'.'||
                 STRIP(NAME,T,' ')||' INDEX(ALL) SHRLEVEL CHANGE'||
                 '                                              '
                ,1,80)
      FROM SYSIBM.SYSTABLESPACE
  WHERE
    (DBNAME IN ('DBTEST1','DBTEST1A')
     AND
     SUBSTR(CHAR(CURRENT_TIMESTAMP),9,2)
```



```
      IN ('Ø1','Ø2','Ø3','Ø4','Ø5','Ø6','Ø7'))
  OR
    (DBNAME IN ('DBTEST2')
     AND
     SUBSTR(CHAR(CURRENT_TIMESTAMP),9,2)
      IN ('Ø8','Ø9','1Ø','11','12','13','14'))
  OR
    (DBNAME IN ('DBTEST3')
     AND
     SUBSTR(CHAR(CURRENT_TIMESTAMP),9,2)
      IN ('15','16','17','18','19','2Ø','21'))
  OR
    (DBNAME IN ('DBTEST4','DBTEST5','DBTEST5A')
     AND
     SUBSTR(CHAR(CURRENT_TIMESTAMP),9,2)
      IN ('22','23','24','25','26','27','28'))
  OR
    (DBNAME IN ('DSNDBØ6')
     AND
     SUBSTR(CHAR(CURRENT_TIMESTAMP),9,2)
       IN ('29','3Ø','31'))
  OR
    (DBNAME IN (SELECT NAME FROM SYSIBM.SYSDATABASE
                WHERE STGROUP = 'main STOGROUP')
     AND DBNAME NOT IN
         ('DBTEST1','DBTEST1A','DBTEST2', 'DBTEST3',
        'DBTEST4','DBTEST5','DBTEST5A')
     AND
     SUBSTR(CHAR(CURRENT_TIMESTAMP),9,2)
       IN ('29','3Ø','31'))
  WITH UR;
/*
//*
//*     LIST RUNSTATS INPUT DECK
//*
//STEPØØ2Ø EXEC PGM=IEBGENER,COND=(5,LT)
//SYSIN DD DUMMY
//SYSPRINT DD SYSOUT=*
//SYSUT2 DD SYSOUT=*
//SYSUT1 DD DISP=(OLD,PASS),DSN=&&RECØØ
//*
//*     UPDATE RUN STATS FOR SELECTED DB2 TABLESPACES
//*
//STEPØØ3Ø EXEC DSNUPROC,SYSTEM=subsystem,UID='utilid',UTPROC='',
//               COND=(5,LT)
//DSNUPROC.SYSIN DD DISP=(OLD,DELETE),DSN=&&RECØØ
//*
```

This method is easily used to produce input for utilities where all the required input can be contained on one line. It can, however, be extended with UNIONs and ORDER BY to produce multi-line input decks. Here is an example of a SELECT that will generate a multi-line RUNSTATS input deck. The SELECT again forces DSNTIAUL to generate 80-byte records, but this time utilizing positions 73 through 80 to store identifiers (DBID/OBID/sequence number), which can be used in an ORDER BY to ensure the records are generated in the correct sequence:

```
SELECT SUBSTR(
               'RUNSTATS TABLESPACE '||
                STRIP(DBNAME,T,' ')||'.'||
                STRIP(NAME,T,' ')||' '||
                '                                                '||
                '                                                '
        ,1,72),
        SUBSTR(
               SUBSTR(HEX(DBID),2,3)||
               SUBSTR(HEX(OBID),1,4)||'1'
        ,1,8)
     FROM SYSIBM.SYSTABLESPACE
  WHERE
     DBNAME IN ('DBP','DBT')
  UNION ALL
  SELECT SUBSTR(
               '     '||
               'INDEX(ALL) SHRLEVEL REFERENCE'||
                '                                               '||
                '                                               '
         ,1,72),
        SUBSTR(
               SUBSTR(HEX(DBID),2,3)||
               SUBSTR(HEX(OBID),1,4)||'2'
         ,1,8)
     FROM SYSIBM.SYSTABLESPACE
  WHERE
     DBNAME IN ('DBP','DBT')
  ORDER BY 2
  WITH UR;
```

Note the use of the SUBSTR function to generate two CHAR columns in the result set – if this was omitted the columns would be of type VARCHAR and would not form valid DSNUTIL input statements.

GENERATING INPUT FOR DSN COMMANDS

This approach can also be used to generate input for DSN commands – BIND PACKAGE being one obvious example. The following job will rebind a selected group of packages with EXPLAIN(YES) – the DBA can determine, through the SELECT statement used in the DSNTIAUL step, exactly which packages to rebind, without requiring a full list. He is not restricted to either naming all the packages or rebinding a whole collection, just to get EXPLAIN output for some.

```
//xxxxxx JOB
//*
//*  THIS JOB BUILDS REBIND INPUT IN TIAUL
//*  LISTS IT OUT
//*  AND RUNS IT
//*
//JOBLIB    DD DSN=DSN.SDSNLOAD,DISP=SHR
//*
//*        RUN UNLOAD PROGRAM
//*
//GENERATE EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(subsystem)
 RUN  PROGRAM(DSNTIAUL) PLAN(DSNTIAUL) -
      LIB('db2.RUNLIB.LOAD') PARM('SQL')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSREC00 DD UNIT=WORK,DSN=&&REC00,DISP=(NEW,PASS),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
//SYSPUNCH DD SYSOUT=*
//SYSIN    DD  *
  SELECT SUBSTR(
                'REBIND PACKAGE('||
                 STRIP(COLLID,T,' ')||'.'||
                 STRIP(NAME,T,' ')||')'||
                 ' EXPLAIN(YES)                                '||
                 '                                             '
                ,1,80)
      FROM SYSIBM.SYSPACKAGE
  WHERE
     COLLID IN ('COLL1')
     AND NAME LIKE 'APP1%'
  WITH UR;
/*
//*
//* LIST INPUT
//*
```

```
//INPUT EXEC PGM=IEBGENER
//SYSIN DD DUMMY
//SYSPRINT DD SYSOUT=*
//SYSUT2 DD SYSOUT=*
//SYSUT1 DD DISP=(OLD,PASS),DSN=&&REC00
//*
//* REBIND
//*
//REBIND   EXEC PGM=IKJEFT01,
//         DYNAMNBR=20,
//         COND=(4,LT)
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2T)
//         DD  DISP=(OLD,DELETE),DSN=&&REC00
//         DD  *
  END
/*
//* EOJ
//
```

FURTHER THOUGHTS

The examples I have included have all been actively used under DB2 Version 5. With Version 6 now becoming more generally available with more complex character manipulation functions, it will be possible to generate even more sophisticated SELECT statements to use with DSNTIAUL in this way. Indeed, in the not very distant future it may be possible to generate all the DDL, DSN, and utility statements needed by a DBA using this type of approach – it may be that, as is now the case in the Oracle world, DB2 DBAs will maintain a standard set of SQL scripts to perform all the regular housekeeping functions, rather than maintaining fixed decks for such jobs or relying on third-party tools.

Phil Button
Consultant (UK)

Cleaning up the SYSIBM.SYSCOPY tablespace

This is an update to DB2MODRC, written by Jeffrey H Nix of AutoZone Inc, which was published in *DB2 Update* in Issue 11 (September 1993).

In the original article the rationale for this routine is explained as follows: “Removing old image copies from SYSIBM.SYSCOPY using the MODIFY RECOVERY command is a manual process and can be very unforgiving if a slip is made while entering the date part of the command. However, I have written a program that will use the MVS catalog(s) to help me determine when to delete image copies without having to worry about the consequences of making a mistake. Essentially, once a dataset has been uncatalogued and deleted from MVS, it is no longer available for DB2 recovery purposes, and it can be deleted from SYSIBM.SYSCOPY.”

The changes from the original in this version are:

- It supports ICs for partitioned tablespaces.
- It accommodates the year 2000.
- It generates the input parameters to DSNUPROC, rather than the JCL to run DSNUPROC, and input.
- The code has been restructured for readability.
- There is error reporting of SQL errors.
- There have been minor report changes.

The run unit consists of DB2MODRC, $CATSCAN, and $DB2TRM.

Input:

- DB2 sub-system ID.
- An instream indicator (Y=instream JCL to process statements, N=generate JCL to execute modify recovery utility).

Output:

- Modify recovery parameter statements, and execution JCL if instream = N.
- Reports.

The DB2MODRC (mainline) program uses two different cursors to read the SYSIBM.SYSCOPY (syscopy) table.The cursor CMODRC1 selects each row from SYSCOPY where the ICTYPE = 'F' by database name, tablespace name, dataset number (partition number or zero for all or non-partitioned), image copy date, and image copy time.

Each row from this cursor is fetched, and it will determine whether the database tablespace has changed:

- If the object has changed, check if any back-ups have been found. If no back-ups are found for the tablespace, it generates a report record and processes the new database tablespace.
- If the object has not changed, processing continues.

Set the flag to indicate that no back-ups exist and call $CATSCAN to search the OS CATALOG to determine whether the back-up dataset does exist. If it is not found, set the flag to generate the ‘Modify Recovery’ record and generate the report record.

If it is found we do the following: clear the flag and build the MODIFY RECOVERY command to remove the image copy entries prior to this (the last good) image copy. If we do not get any hits for a specific tablespace, then we must delete all but the last copy for that tablespace/ partition.

This is so we do not cause the status of the tablespace to become ‘COPY PENDING’, and then we print it on the NOBACKUPS report. At the same time we must set a flag to determine if all copies are good so that we do not build a MODIFY RECOVERY command.

The cursor CMODRC2 selects each row from SYSCOPY where the ICTYPE = 'F', database name = ‘saved database name’, tablespace name = ‘saved tablespace name’, dataset number = ‘saved dataset number’, and image copy date and time is less than the oldest existing image copy’s date and time. The rows are fetched from this cursor to list the rows that are to be removed from SYSCOPY.

The $DB2TRM routine is a PL/I routine to invoke IBM-supplied routine DSNTIAR and format error messages. Your installation probably has a similar routine – it is included for clarity.

DB2MODRC

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. DB2MODRC.
*****************************************************************
* MAIN-ROUTINE: DB2MODRC.                                       *
* PURPOSE: Clean-up SYSIBM.SYSCOPY.                             *
* HOW: Via the MODIFY RECOVERY command.                         *
* PARMS PASSED: "DB2 SUBSYSTEM-ID", "INSTREAM INDICATOR (N)"    *
* RETURN CODES:                                                 *
*  Ø = SUCCESSFUL                                               *
*  4 = SUCCESSFUL                                               *
* 16 = SEVERE ERROR                                             *
* INPUT / OUTPUT DATA:                                          *
*    I  -  SYSIBM.SYSCOPY                                       *
*    O  -  MODIFY.RECOVERY.JCL                                  *
*    O  -  DELETES REPORT, NO BACK-UP REPORT                    *
*****************************************************************
* CALLED-ROUTINE: $CATSCAN                                      *
* PURPOSE: Searching the MVS CATALOGS                           *
* HOW: Via the "LOCATE" and "CAMLST" macro instructions         *
* PARMS PASSED: "DSNAME"                                        *
* RETURN CODES:                                                 *
*  Ø = SUCCESSFUL                                               *
*  4 = DSNAME NOT FOUND                                         *
*  8 = DSNAME NOT PASSED                                        *
* 12 = DSNAME INVALID                                           *
*****************************************************************
* CALLED-ROUTINE: $DB2RRC                                       *
* PURPOSE: Generate DB2 error message text                      *
* HOW: Invoke DSNTAIR routine and formats message               *
* PARMS PASSED: "SQLCA", "MTOPCB", "CALLID"                     *
* RETURN CODES:                                                 *
*  N/A - Abends run unit                                        *
*****************************************************************
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT MODRCJCL  ASSIGN TO  MODRCJCL.
     SELECT REPORT1   ASSIGN TO  REPORT1.
     SELECT REPORT2   ASSIGN TO  REPORT2.
 DATA DIVISION.
 FILE SECTION.
```

```
*                       J C L                                   *
 FD  MODRCJCL
     RECORDING MODE IS F
     LABEL RECORDS ARE STANDARD
     BLOCK CONTAINS Ø RECORDS
     DATA RECORD IS MODRCJCL-REC.
 Ø1  MODRCJCL-REC               PIC  X(8Ø).
*                     R E P O R T S                             *
 FD  REPORT1
     LABEL RECORDS ARE STANDARD
     BLOCK CONTAINS Ø RECORDS
     RECORDING MODE IS F
     DATA RECORD IS REPT1-REC.
 Ø1  REPT1-REC                  PIC X(133).
 FD  REPORT2
     LABEL RECORDS ARE STANDARD
     BLOCK CONTAINS Ø RECORDS
     RECORDING MODE IS F
     DATA RECORD IS REPT2-REC.
 Ø1  REPT2-REC                  PIC X(133).
*               W O R K I N G - S T O R A G E                   *
 WORKING-STORAGE SECTION.
 Ø1  WS-SQLCODE                 PIC  9(1Ø) VALUE Ø.
 Ø1  WS-SQL-ERROR               PIC  X(5Ø) VALUE SPACES.
 Ø1  MODREC-FLAG                PIC  X(Ø1) VALUE 'N'.
 Ø1  NOBACKUPS-FLAG             PIC  X(Ø1) VALUE 'Y'.
 Ø1  REPT1-TOTAL                PIC S9(Ø5) COMP SYNC VALUE +Ø.
 Ø1  REPT2-TOTAL                PIC S9(Ø5) COMP SYNC VALUE +Ø.
 Ø1  REPT1-LINECNT              PIC S9(Ø3) COMP-3 VALUE 56.
 Ø1  REPT2-LINECNT              PIC S9(Ø3) COMP-3 VALUE 56.
 Ø1  REPT1-PAGECNT              PIC S9(Ø5) COMP-3 VALUE ZERO.
 Ø1  REPT2-PAGECNT              PIC S9(Ø5) COMP-3 VALUE ZERO.
 77  POS2-PTR                  PIC S9(Ø3) COMP-3 VALUE ZERO.
 77  POS3-PTR                  PIC S9(Ø3) COMP-3 VALUE ZERO.
 77  ABEND-CODE                PIC S9(Ø5) COMP SYNC VALUE ZERO.
 77  MTO-PCB                   POINTER    SYNC.
 77  RUN-DATE                  PIC 9(Ø6).
 Ø1  CALL-ID SYNC.
     Ø5  FILLER                 PIC X(Ø5) VALUE 'UTMR-'.
     Ø5  CALL-NUM               PIC X(Ø2).
 Ø1  DB2-CONNECT-DISCONNECT-PARMS.
     Ø5  OPEN-CA                PIC X(12) VALUE 'OPEN        '.
     Ø5  CLOSE-CA               PIC X(12) VALUE 'CLOSE       '.
     Ø5  TRANSLATE-CA           PIC X(12) VALUE 'TRANSLATE   '.
     Ø5  TERM-OP                PIC X(Ø4) VALUE 'SYNC'.
     Ø5  DB2ID                  PIC X(4)  VALUE 'DSN '.
     Ø5  DB2PLAN                PIC X(8)  VALUE 'UTØ1MRØ1'.
 Ø1  WS-MODREC-COMMAND          PIC X(8Ø).
 Ø1  WORK-AREA.
     Ø5 PREV-DBTS.
```

```
           10 PREV-DBNAME         PIC  X(08) VALUE SPACES.
           10 PREV-TSNAME         PIC  X(08) VALUE SPACES.
       05 HOLD-DBTS.
           10 HOLD-DBNAME         PIC  X(08) VALUE SPACES.
           10 HOLD-TSNAME         PIC  X(08) VALUE SPACES.
       05 WS-DBNAME               PIC  X(08) VALUE SPACES.
       05 WS-TSNAME               PIC  X(08) VALUE SPACES.
       05 WS-DSNUM                PIC  S9(09) COMP  VALUE +0.
       05 WS-DSNAME               PIC  X(44) VALUE SPACES.
       05 WS-ICDATE               PIC  X(06) VALUE SPACES.
       05 WS-TIMESTAMP            PIC  X(26) VALUE SPACES.
       05 WS-COUNT                PIC  S9(8) COMP  VALUE ZERO.
       05 WS-DBNAME-TSNAME        PIC  X(17) VALUE SPACES.
       05 WS-DSNUM-DISPLAY        PIC  Z(03)9 VALUE ZERO.
       05 WS-DSNUM-DISPLAY-X      REDEFINES WS-DSNUM-DISPLAY
                                  PIC  X(04).
* D B 2   D A T A   A R E A S               *
     EXEC SQL INCLUDE SQLCA  END-EXEC.
*        S Y S I B M . S Y S C O P Y  *
     EXEC SQL INCLUDE SYSCOPY END-EXEC.
* ALL FULL COPIES IN DBNAME, TSNAME, DSNUM, TIMESTAMP SEQUENCE *
     EXEC SQL
          DECLARE CMODRC1 CURSOR FOR
                  SELECT DBNAME,
                         TSNAME,
                         DSNUM ,
                         DSNAME,
                         TIMESTAMP,
                         ICDATE,
                         ICTIME
                  FROM SYSIBM.SYSCOPY
                  WHERE ICTYPE = 'F'
                  ORDER BY DBNAME, TSNAME, DSNUM, TIMESTAMP
          END-EXEC.
* ALL FULL COPIES FOR A SPECIFIC DBNAME,TSNAME, DSNUM BEFORE    *
* DATE PORTION FROM TIMESTAMP -- ICDATE DOESN'T HAVE CENTURY    *
* FULL TIMESTAMP ISN'T USED BECAUSE MODIFY USES ONLY DATE VALUE *
     EXEC SQL
          DECLARE CMODRC2 CURSOR FOR
                  SELECT DBNAME,
                         TSNAME,
                         DSNUM ,
                         DSNAME,
                         TIMESTAMP,
                         ICDATE,
                         ICTIME
                  FROM SYSIBM.SYSCOPY
                  WHERE DBNAME = :WS-DBNAME
                    AND TSNAME = :WS-TSNAME
                    AND DSNUM  = :WS-DSNUM
```

```
                    AND DATE(TIMESTAMP) <
                            DATE(SUBSTR(:WS-TIMESTAMP,1,1Ø))
                    AND ICTYPE = 'F'
                  ORDER BY TIMESTAMP
           END-EXEC.
*             M V S   J C L                *
 Ø1  JOBCARD.
     Ø2  FILLER           PIC X(24) VALUE
         '//DB2MODRC JOB (),'.
     Ø2  FILLER           PIC X(46) VALUE
         'CLASS=D,MSGCLASS=X,NOTIFY=&SYSUID'.
     Ø2  FILLER           PIC X(24) VALUE SPACES.
 Ø1  EXECUTE-STMT.
     Ø2  FILLER           PIC X(45) VALUE
         '//MODIFYRC EXEC DSNUPROC,UID=DB2MODRC,SYSTEM='.
     Ø2  SYSID            PIC X(Ø4) VALUE SPACES.
     Ø2  FILLER           PIC X(31) VALUE SPACES.
 Ø1  SYSIN-OVERRIDE.
     Ø2  FILLER           PIC X(21) VALUE
         '//DSNUPROC.SYSIN DD *'.
     Ø2  FILLER           PIC X(59) VALUE SPACES.
*          R E P T 1 - T I T L E            *
 Ø1  REPT-TITLE.
     Ø2  FILLER           PIC X(Ø7) VALUE
         ' SSID: '.
     Ø2  REPT-SYSID       PIC X(Ø4) VALUE SPACES.
     Ø2  FILLER           PIC X(32) VALUE SPACES.
     Ø2  FILLER           PIC X(47) VALUE
         'Y O U R   C O M P A N Y   N A M E   H E R E |'.
     Ø2  FILLER           PIC X(33) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø6) VALUE 'PAGE: '.
     Ø2  REPT-PAGECNT     PIC 9(Ø4) VALUE ZEROES.
     Ø2  FILLER           PIC X(Ø1) VALUE SPACES.
*          R E P T 1 - T I T L E            *
 Ø1  REPT1-TITLE.
     Ø2  FILLER           PIC X(22) VALUE
         ' PGM: DB2MODRC-Ø1'.
     Ø2  FILLER           PIC X(83) VALUE
         ' TABLESPACE IMAGECOPIES NOT CATALOGUED - SELECTED TO BE
-        'REMOVED FROM SYSIBM.SYSCOPY'.
     Ø2  FILLER           PIC X(14) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø6) VALUE 'DATE: '.
     Ø2  REPT1-DATE       PIC 99/99/99 VALUE ZEROES.
     Ø2  FILLER           PIC X(Ø1) VALUE SPACES.
*          R E P T 1 - H E A D              *
 Ø1  REPT1-HEAD.
     Ø2  FILLER           PIC X(2Ø) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø6) VALUE 'DBNAME'.
     Ø2  FILLER           PIC X(Ø5) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø6) VALUE 'TSNAME'.
```

```
     Ø2  FILLER           PIC X(Ø8) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø5) VALUE 'DSNUM'.
     Ø2  FILLER           PIC X(Ø5) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø6) VALUE 'ICDATE'.
     Ø2  FILLER           PIC X(Ø5) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø6) VALUE 'DSNAME'.
     Ø2  FILLER           PIC X(48) VALUE SPACES.
*          R E P T 1 - D A S H              *
 Ø1  REPT1-DASH.
     Ø2  FILLER           PIC X(2Ø) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø6) VALUE ALL '-'.
     Ø2  FILLER           PIC X(Ø5) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø6) VALUE ALL '-'.
     Ø2  FILLER           PIC X(Ø8) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø5) VALUE ALL '-'.
     Ø2  FILLER           PIC X(Ø5) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø6) VALUE ALL '-'.
     Ø2  FILLER           PIC X(Ø5) VALUE SPACES.
     Ø2  FILLER           PIC X(44) VALUE ALL '-'.
     Ø2  FILLER           PIC X(22) VALUE SPACES.
*          R E P T 1 - D E T A I L          *
 Ø1  REPT1-DETAIL.
     Ø2  FILLER           PIC X(2Ø) VALUE SPACES.
     Ø2  REPT1-DBNAME     PIC X(Ø8) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø3) VALUE SPACES.
     Ø2  REPT1-TSNAME     PIC X(Ø8) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø6) VALUE SPACES.
     Ø2  REPT1-DSNUM      PIC X(Ø4) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø6) VALUE SPACES.
     Ø2  REPT1-ICDATE     PIC X(Ø6) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø5) VALUE SPACES.
     Ø2  REPT1-DSNAME     PIC X(44) VALUE SPACES.
     Ø2  FILLER           PIC X(2Ø) VALUE SPACES.
*          R E P T 2 - T I T L E            *
 Ø1  REPT2-TITLE.
     Ø2  FILLER           PIC X(3Ø) VALUE
         ' PGM: DB2MODRC-Ø2'.
     Ø2  FILLER           PIC X(67) VALUE
         'TABLESPACES WITHOUT IMAGECOPIES - NOT REMOVED TO AVOID C
-        'OPY PENDING'.
     Ø2  FILLER           PIC X(22) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø6) VALUE 'DATE: '.
     Ø2  REPT2-DATE       PIC 99/99/99 VALUE ZEROES.
     Ø2  FILLER           PIC X(Ø1) VALUE SPACES.
*          R E P T 2 - H E A D              *
 Ø1  REPT2-HEAD.
     Ø2  FILLER           PIC X(2Ø) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø6) VALUE 'DBNAME'.
     Ø2  FILLER           PIC X(Ø5) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø6) VALUE 'TSNAME'.
```

```
     Ø2  FILLER           PIC X(Ø8) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø5) VALUE 'DSNUM'.
     Ø2  FILLER           PIC X(Ø5) VALUE SPACES.
     Ø2  FILLER           PIC X(11) VALUE 'LAST-ICDATE'.
     Ø2  FILLER           PIC X(Ø5) VALUE SPACES.
     Ø2  FILLER           PIC X(13) VALUE 'LAST-ICDSNAME'.
     Ø2  FILLER           PIC X(56) VALUE SPACES.
*         R E P T 2 - D A S H              *
 Ø1  REPT2-DASH.
     Ø2  FILLER           PIC X(2Ø) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø6) VALUE ALL '-'.
     Ø2  FILLER           PIC X(Ø5) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø6) VALUE ALL '-'.
     Ø2  FILLER           PIC X(Ø8) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø5) VALUE ALL '-'.
     Ø2  FILLER           PIC X(Ø5) VALUE SPACES.
     Ø2  FILLER           PIC X(11) VALUE ALL '-'.
     Ø2  FILLER           PIC X(Ø5) VALUE SPACES.
     Ø2  FILLER           PIC X(44) VALUE ALL '-'.
     Ø2  FILLER           PIC X(2Ø) VALUE SPACES.
*         R E P T 2 - D E T A I L          *
 Ø1  REPT2-DETAIL.
     Ø2  FILLER           PIC X(2Ø) VALUE SPACES.
     Ø2  REPT2-DBNAME     PIC X(Ø8) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø3) VALUE SPACES.
     Ø2  REPT2-TSNAME     PIC X(Ø8) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø6) VALUE SPACES.
     Ø2  REPT2-DSNUM      PIC X(Ø4) VALUE SPACES.
     Ø2  FILLER           PIC X(Ø6) VALUE SPACES.
     Ø2  REPT2-ICDATE     PIC X(Ø6) VALUE SPACES.
     Ø2  FILLER           PIC X(1Ø) VALUE SPACES.
     Ø2  REPT2-DSNAME     PIC X(44) VALUE SPACES.
     Ø2  FILLER           PIC X(2Ø) VALUE SPACES.
*             L I N K A G E   S E C T I O N                *
 LINKAGE SECTION.
 Ø1  CAFPARMS.
     Ø5  LENGTH-OF-PARM   PIC S9(4) COMP.
     Ø5  PSSID            PIC X(4).
     Ø5  INSTREAM-IND     PIC X.
/*       P R O C E D U R E   D I V I S I O N                *
 PROCEDURE DIVISION USING CAFPARMS.
     IF  LENGTH-OF-PARM        LESS +4
         DISPLAY 'PARAMETER MISSING ON EXEC - RUN ABORTED'
         MOVE +255             TO  ABEND-CODE
         CALL 'ILBOABNØ'     USING  ABEND-CODE
     END-IF
     PERFORM 9Ø1Ø-INITIALIZE THRU 9Ø1Ø-EXIT
     PERFORM Ø1ØØ-MAINLINE THRU Ø1ØØ-EXIT
     PERFORM 99ØØ-TERMINATE THRU 99ØØ-EXIT
     GOBACK
```

```
           CONTINUE.
      /*
       Ø1ØØ-MAINLINE.
      * Open CURSOR containing all full copies by DBNAME,TSNAME,ICDATE*
           EXEC SQL OPEN CMODRC1 END-EXEC
           IF  SQLCODE        EQUAL  ZERO
               PERFORM Ø25Ø-FETCH-CMODRC1 THRU Ø25Ø-EXIT
               IF    LENGTH-OF-PARM   LESS   5
                 OR  INSTREAM-IND     EQUAL 'N'
                   WRITE MODRCJCL-REC FROM JOBCARD
                   INITIALIZE MODRCJCL-REC
                   WRITE MODRCJCL-REC FROM EXECUTE-STMT
                   INITIALIZE MODRCJCL-REC
                   WRITE MODRCJCL-REC FROM SYSIN-OVERRIDE
               END-IF
               MOVE DBNAME TO PREV-DBNAME
               MOVE TSNAME TO PREV-TSNAME
             ELSE
               MOVE 'OPEN ERROR ON CMODRC1' TO WS-SQL-ERROR
               MOVE 'Ø1'                 TO  CALL-NUM
               PERFORM 9Ø2Ø-SQL-ERROR THRU 9Ø2Ø-EXIT
           END-IF
           PERFORM Ø2ØØ-PROCESS-SYSCOPY THRU Ø2ØØ-EXIT
               UNTIL  SQLCODE  NOT EQUAL ZERO
                  OR  RETURN-CODE  EQUAL 16
           EVALUATE SQLCODE
               WHEN Ø
                    CONTINUE
               WHEN +1ØØ
                    EXEC SQL CLOSE CMODRC1 END-EXEC
                    IF SQLCODE NOT = Ø
                       MOVE 'CLOSE ERROR ON CMODRC1' TO WS-SQL-ERROR
                       MOVE 'Ø2'                  TO  CALL-NUM
                       PERFORM 9Ø2Ø-SQL-ERROR THRU 9Ø2Ø-EXIT
XIT
                    END-IF
               WHEN OTHER
                    MOVE 'SQL ERROR ON CMODRC1' TO WS-SQL-ERROR
                    MOVE 'Ø3'             TO  CALL-NUM
                    PERFORM 9Ø2Ø-SQL-ERROR THRU 9Ø2Ø-EXIT
           END-EVALUATE
           CONTINUE.
       Ø1ØØ-EXIT.
           EXIT.
      / * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      * Fetch from cursor one, if it's the first row then build JCL   *
      * and set some switches, else go search the MVS CATALOG. If no  *
      * hit then fetch another row. Once we get a hit we must build   *
      * the MODIFY RECOVERY command and print what's going to be      *
      * deleted and go on to the next TABLESPACE. If we DO NOT get any*
```

```
* hits for a specific TABLESPACE then we must delete all but    *
* the last copy for that tablespace. This is so we DO NOT cause *
* the status of the TABLESPACE to become "COPY PENDING", and    *
* then we print it on the NOBACKUPS report. At the same time we *
* must set a flag to determine if all copies are good so that   *
* we DO NOT build  MODIFY RECOVERY command.                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 Ø2ØØ-PROCESS-SYSCOPY.
     IF      TSNAME      EQUAL  HOLD-TSNAME
       AND   DBNAME      EQUAL  HOLD-DBNAME
         CONTINUE
       ELSE
         IF    TSNAME    EQUAL  PREV-TSNAME
           AND DBNAME    EQUAL  PREV-DBNAME
             CONTINUE
           ELSE
             IF NOBACKUPS-FLAG = 'Y'
                  MOVE 'N' TO NOBACKUPS-FLAG
                  PERFORM 2ØØØ-PRINT-REPORT2 THRU 2ØØØ-EXIT
                  PERFORM Ø5ØØ-BUILD-MODREC THRU Ø5ØØ-EXIT
             END-IF
             MOVE TSNAME TO PREV-TSNAME
             MOVE DBNAME TO PREV-DBNAME
         END-IF
         MOVE TSNAME TO WS-TSNAME
         MOVE DBNAME TO WS-DBNAME
         MOVE ICDATE TO WS-ICDATE
         MOVE DSNAME TO WS-DSNAME
         MOVE DSNUM  TO WS-DSNUM
         MOVE TIMESTAMP TO WS-TIMESTAMP
         IF  DSNUM  = Ø
             MOVE 'ALL'         TO  WS-DSNUM-DISPLAY-X
           ELSE
             MOVE DSNUM         TO  WS-DSNUM-DISPLAY
         END-IF
         MOVE 'Y' TO NOBACKUPS-FLAG
         PERFORM Ø3ØØ-CHECK-CATALOG THRU Ø3ØØ-EXIT
     END-IF
     PERFORM Ø25Ø-FETCH-CMODRC1 THRU Ø25Ø-EXIT
     CONTINUE.
 Ø2ØØ-EXIT.
     EXIT.
/
 Ø25Ø-FETCH-CMODRC1.
     EXEC SQL FETCH CMODRC1
          INTO :DBNAME,
             :TSNAME,
             :DSNUM ,
             :DSNAME,
             :TIMESTAMP,
```

```
             :ICDATE,
             :ICTIME
     END-EXEC.
 Ø25Ø-EXIT.
     EXIT.
/
 Ø3ØØ-CHECK-CATALOG.
* Routine used to search MVS CATALOG(s) for a specific dsname.  *
*                                                              *
*   RC = Ø - CATALOGUED.                                       *
*   RC = 4 - NOT CATALOGUED.                                   *
*                                                              *
     CALL '$CATSCAN' USING WS-DSNAME.
     EVALUATE RETURN-CODE
         WHEN Ø     MOVE 'N' TO NOBACKUPS-FLAG
                    PERFORM Ø5ØØ-BUILD-MODREC THRU Ø5ØØ-EXIT
                    PERFORM 1ØØØ-PRINT-REPORT1 THRU 1ØØØ-EXIT
         WHEN 4     MOVE 'Y' TO MODREC-FLAG
         WHEN 8     DISPLAY ' CATSCAN ERROR: NO DSNAME PASSED '
         WHEN 12    DISPLAY ' CATSCAN ERROR: INVALID DSNAME '
         WHEN OTHER
                 DISPLAY 'CATSCAN ERROR: RETURN-CODE = '
                     RETURN-CODE
                     'MVS-DSNAME: ' WS-DSNAME
                 DISPLAY 'NOTIFY YOUR DB2 SYSTEM ADMINISTRATOR'
                 MOVE 16 TO RETURN-CODE
     END-EVALUATE
     CONTINUE.
 Ø3ØØ-EXIT.
     EXIT.
/
* First SELECT COUNT(*) to see if there really is any to delete.*
* Routine used to build MODIFY RECOVERY command.               *
*                                                              *
* Modify recovery permits only date values. So two image copies *
* taken the same day with one still being catalogued cannot    *
* be modified .                                                *
*                                                              *
 Ø5ØØ-BUILD-MODREC.
     IF MODREC-FLAG = 'Y'
        EXEC SQL SELECT COUNT(*)
           INTO :WS-COUNT
           FROM SYSIBM.SYSCOPY
          WHERE DBNAME = :WS-DBNAME
            AND TSNAME = :WS-TSNAME
            AND DSNUM  = :WS-DSNUM
            AND DATE(TIMESTAMP) < DATE(SUBSTR(:WS-TIMESTAMP,1,1Ø))
            AND ICTYPE = 'F'
        END-EXEC
        IF SQLCODE NOT = Ø
```

```
            MOVE 'SQL ERROR ON SELECT ' TO WS-SQL-ERROR
            MOVE 'Ø4'                   TO  CALL-NUM
            PERFORM 9Ø2Ø-SQL-ERROR THRU 9Ø2Ø-EXIT
         END-IF
         IF WS-COUNT > Ø
            MOVE 'N' TO MODREC-FLAG
            INITIALIZE WS-DBNAME-TSNAME POS2-PTR POS3-PTR
            STRING WS-DBNAME '.' WS-TSNAME
                DELIMITED BY SPACES INTO WS-DBNAME-TSNAME
            END-STRING
            MOVE ZERO            TO  POS2-PTR POS3-PTR
            INSPECT WS-DSNUM-DISPLAY-X
               TALLYING POS2-PTR
               FOR LEADING SPACES
            INSPECT WS-DSNUM-DISPLAY-X
               TALLYING POS3-PTR
               FOR ALL SPACES
            INITIALIZE WS-MODREC-COMMAND, MODRCJCL-REC
            STRING ' MODIFY RECOVERY TABLESPACE '
                WS-DBNAME-TSNAME
                DELIMITED BY HIGH-VALUES INTO WS-MODREC-COMMAND
            END-STRING
            WRITE MODRCJCL-REC FROM WS-MODREC-COMMAND
            INITIALIZE WS-MODREC-COMMAND, MODRCJCL-REC
            STRING
                ' DSNUM '
                WS-DSNUM-DISPLAY-X(POS2-PTR + 1:4 - POS3-PTR)
                ' DELETE DATE(' WS-ICDATE ')'
                DELIMITED BY HIGH-VALUES INTO WS-MODREC-COMMAND
            END-STRING
            WRITE MODRCJCL-REC FROM WS-MODREC-COMMAND
         END-IF
      END-IF
      MOVE WS-TSNAME TO HOLD-TSNAME.
      MOVE WS-DBNAME TO HOLD-DBNAME.
 Ø5ØØ-EXIT.
      EXIT.
/
* Routine used to print the names of the copies that will be    *
* deleted whenever the MODIFY RECOVERY command executes.        *
 1ØØØ-PRINT-REPORT1.
      EXEC SQL OPEN CMODRC2 END-EXEC
      IF  SQLCODE            EQUAL ZERO
          PERFORM 12ØØ-FETCH-CMODRC2 THRU 12ØØ-EXIT
        ELSE
          MOVE 'OPEN ERROR ON CMODRC2' TO WS-SQL-ERROR
          MOVE 'Ø5'          TO  CALL-NUM
          PERFORM 9Ø2Ø-SQL-ERROR THRU 9Ø2Ø-EXIT
      END-IF
      PERFORM 11ØØ-SETUP-REPORT1 THRU 11ØØ-EXIT
```

```
           UNTIL  SQLCODE  NOT EQUAL ZERO
         EVALUATE SQLCODE
             WHEN +100
                  EXEC SQL CLOSE CMODRC2 END-EXEC
                  IF SQLCODE NOT = 0
                     MOVE 'CLOSE ERROR ON CMODRC2' TO WS-SQL-ERROR
                     MOVE '06'             TO  CALL-NUM
                     PERFORM 9020-SQL-ERROR THRU 9020-EXIT
                  END-IF
             WHEN OTHER
                  MOVE 'SQL ERROR ON CMODRC2' TO WS-SQL-ERROR
                  MOVE '07'             TO  CALL-NUM
                  PERFORM 9020-SQL-ERROR THRU 9020-EXIT
         END-EVALUATE
         CONTINUE.
      1000-EXIT.
         EXIT.
     /
      1100-SETUP-REPORT1.
     * Fetch rows that will be deleted by MODIFY RECOVERY command.   *
         IF REPT1-LINECNT > 55
            ADD 1 REPT1-PAGECNT  GIVING REPT1-PAGECNT REPT-PAGECNT
            WRITE REPT1-REC FROM REPT-TITLE AFTER ADVANCING PAGE
            WRITE REPT1-REC FROM REPT1-TITLE AFTER ADVANCING 1
            WRITE REPT1-REC FROM REPT1-HEAD  AFTER ADVANCING 3 LINES
            WRITE REPT1-REC FROM REPT1-DASH  AFTER ADVANCING 1 LINES
            MOVE SPACES TO REPT1-REC
            WRITE REPT1-REC
            MOVE 5 TO REPT1-LINECNT
         END-IF
         MOVE DBNAME TO REPT1-DBNAME.
         MOVE TSNAME TO REPT1-TSNAME.
         MOVE ICDATE TO REPT1-ICDATE.
         MOVE DSNAME TO REPT1-DSNAME.
         IF  DSNUM  = 0
             MOVE 'ALL'          TO  WS-DSNUM-DISPLAY-X
           ELSE
             MOVE DSNUM          TO  WS-DSNUM-DISPLAY
         END-IF
         MOVE WS-DSNUM-DISPLAY-X TO REPT1-DSNUM
         WRITE REPT1-REC FROM REPT1-DETAIL   AFTER ADVANCING 1 LINES
         ADD 1 TO REPT1-TOTAL
         ADD 1 TO REPT1-LINECNT
         INITIALIZE REPT1-DETAIL
         PERFORM 1200-FETCH-CMODRC2 THRU 1200-EXIT
         CONTINUE.
      1100-EXIT.
         EXIT.
     /
      1200-FETCH-CMODRC2.
```

```
     EXEC SQL FETCH CMODRC2
          INTO :DBNAME,
               :TSNAME,
               :DSNUM ,
               :DSNAME,
               :TIMESTAMP,
               :ICDATE,
               :ICTIME
     END-EXEC.
 1200-EXIT.
     EXIT.
* Routine used to print the names of the TABLESPACES that have  *
* no full copy to recover to.                                   *
 2000-PRINT-REPORT2.
     IF REPT2-LINECNT > 55
        ADD 1 REPT2-PAGECNT  GIVING REPT2-PAGECNT REPT-PAGECNT
        WRITE REPT2-REC FROM REPT-TITLE AFTER ADVANCING PAGE
        WRITE REPT2-REC FROM REPT2-TITLE AFTER 1
        WRITE REPT2-REC FROM REPT2-HEAD  AFTER ADVANCING 3 LINES
        WRITE REPT2-REC FROM REPT2-DASH  AFTER ADVANCING 1 LINES
        MOVE SPACES TO REPT2-REC
        WRITE REPT2-REC
        MOVE 5 TO REPT2-LINECNT
     END-IF
     MOVE WS-DBNAME TO REPT2-DBNAME.
     MOVE WS-TSNAME TO REPT2-TSNAME.
     MOVE WS-ICDATE TO REPT2-ICDATE.
     MOVE WS-DSNAME TO REPT2-DSNAME.
     IF  WS-DSNUM  = Ø
         MOVE 'ALL'         TO  WS-DSNUM-DISPLAY-X
       ELSE
         MOVE WS-DSNUM      TO  WS-DSNUM-DISPLAY
     END-IF
     MOVE WS-DSNUM-DISPLAY-X TO REPT2-DSNUM
     WRITE REPT2-REC FROM REPT2-DETAIL   AFTER ADVANCING 1 LINES
     ADD 1 TO REPT2-TOTAL.
     ADD 1 TO REPT2-LINECNT.
     INITIALIZE REPT2-DETAIL.
 2000-EXIT.
     EXIT.
/
 9010-INITIALIZE.
     ACCEPT RUN-DATE FROM DATE
     MOVE  RUN-DATE           TO  REPT1-DATE
                                  REPT2-DATE
* CONNECT TO DB2 SUBSYSTEM THROUGH CALL-ATTACH-FACILITY.       *
     MOVE PSSID TO DB2ID, SYSID, REPT-SYSID
     CALL 'DSNALI' USING OPEN-CA, DB2ID, DB2PLAN
     IF   RETURN-CODE        EQUAL  ZERO
          OPEN OUTPUT        MODRCJCL,
```

```
                              REPORT1,
                              REPORT2
           ELSE
             DISPLAY 'ERROR ON CAF, RC = ' RETURN-CODE
             CALL 'DSNALI' USING TRANSLATE-CA,
                              SQLCA
             MOVE '08'              TO  CALL-NUM
             PERFORM 9020-SQL-ERROR THRU 9020-EXIT
         END-IF
         CONTINUE.
     9010-EXIT.
         EXIT.
    * Error routine used for SQL errors.                          *
     9020-SQL-ERROR.
         MOVE SQLCODE         TO  WS-SQLCODE
         DISPLAY WS-SQL-ERROR ' SQLCODE = ' WS-SQLCODE.
         SET MTO-PCB             TO  NULL
         CALL '$DB2RRC'          USING  SQLCA
                                        MTO-PCB
                                        CALL-ID
         CONTINUE.
     9020-EXIT.
         EXIT.
     9900-TERMINATE.
    * Disconnect from DB2 SUBSYSTEM THROUGH CALL ATTACH FACILITY.  *
         CALL 'DSNALI' USING CLOSE-CA TERM-OP.
    * If no reporting was done, say so.                            *
    * Close JCL and REPORTING files.                               *
         IF REPT1-TOTAL = 0
           ADD 1 REPT1-PAGECNT  GIVING REPT1-PAGECNT REPT-PAGECNT
           WRITE REPT1-REC FROM REPT-TITLE AFTER ADVANCING PAGE
           WRITE REPT1-REC FROM REPT1-TITLE AFTER 1
           INITIALIZE REPT1-REC
           MOVE   '*** NO ROWS WILL BE DELETED THIS RUN ***'
                                       TO REPT1-REC
           WRITE REPT1-REC AFTER ADVANCING 3 LINES
           MOVE 4                      TO RETURN-CODE
         END-IF
         IF REPT2-TOTAL = 0
           ADD 1 REPT2-PAGECNT  GIVING REPT2-PAGECNT REPT-PAGECNT
           WRITE REPT2-REC FROM REPT-TITLE AFTER ADVANCING PAGE
           WRITE REPT2-REC FROM REPT2-TITLE AFTER 1
           INITIALIZE REPT2-REC
           MOVE   '*** THERE ARE NO RECORDS FOR REPORT ***'
                                       TO REPT2-REC
           WRITE REPT2-REC AFTER ADVANCING 3 LINES
         END-IF
         CLOSE MODRCJCL,
               REPORT1,
               REPORT2.
```

```
       9900-EXIT.
           EXIT.
```

$CATSCAN

```
  TITLE '$CATSCAN - SEARCH THE CATALOG FOR AN EXISTING DSNAME'
*********************************************************************/
*    MODULE:  $CATSCAN                                              */
*                                                                   */
*    FUNCTION: THIS MODULE IS USED TO SEARCH THE CATALOG            */
*        FOR THE EXISTENCE OF A DATASET.                            */
*        ENABLE COBOL PROGRAMS TO DETERMINE IF A DATASET IS         */
*        CATALOGUED. INVOKES THE LOCATE MACRO                       */
*                                                                   */
*    INPUT: (1) 44 BYTE FIELD CONTAINING DATASET NAME TO BE         */
*               EVALUATED.                                          */
*                                                                   */
*    USAGE:                                                         */
*           05 WS-DSNAME               PIC  X(44) VALUE SPACES.   */
*           CALL '$CATSCAN' USING WS-DSNAME                         */
*                                                                   */
*    RETURN CODE: TEST USING RETURN-CODE SPECIAL REGISTER           */
*                 0 - CATALOGUED                                    */
*                 4 - NOT CATALOGUED                                */
*                 8 - NO PARMS PASSED                               */
*                12 - DATASET NAME BLANK                            */
*********************************************************************/
$CATSCAN CSECT
*        INITIALIZATION                                             *
         SAVE  (14,12)
         BASR  R12,0
         USING *,R12
         ST    R13,SAVEA+4
         LA    R13,SAVEA
         LR    R2,R1              SAVE R1
*        EXTRACT PARMS PASSED.                                      *
         LTR   R2,R2              ANY PARMS PASSED ?
         BNZ   OK01               SO FAR SO GOOD
         LA    R15,8              SET RC = 8
         B     RETURNE            RETURN TO CALLER
OK01     L     R3,0(R2)           LOAD ADDRESS OF DSNAME
         MVC   DSN,0(R3)          GET DSNAME
         CLC   DSN,BLANKS         IS THE DSNAME BLANK ?
         BNE   OK02               IF NOT PRESS ON
         LA    R15,12             SET RC = 12
         B     RETURNE            RETURN TO CALLER
*        TRY AND LOCATE DATASET IN CATALOG                          *
OK02     SLR   R0,R0              CLEAR R0
         LOCATE LOCLIST           READ CATALOG ENTRY
```

```
         LTR   R15,R15            ANY ERRORS ?
         BZ    OKØ3               IF SO, DATASET WAS NOT FOUND
         LA    R15,4              SET RC = 4
         B     RETURNE            RETURN TO CALLER
OKØ3     LH    R1,CATBLOCK        GET NUMBER OF VOLUMES
         LTR   R1,R1              ARE THERE REALLY ANY ?
         BP    RETURN             THIS SHOULD NOT OCCUR, BUT ...
         LA    R15,4              SET RC = 4
         B     RETURNE            RETURN TO CALLER
RETURN   DS    ØH                 RETURN WITH Ø RC
         LA    R15,Ø              SET RETURN CODE TO Ø
RETURNE  DS    ØH                 LABEL FOR RETURN WITH ERROR
         L     R13,4(R13)         GET OLD SAVE AREA ADDRESS
         ST    R15,16(R13)        SAVE RETURN CODE IN OLD R15
         LM    R14,R12,12(R13)    RESTORE REGS
         BR    R14                RETURN
*        WORK AREAS
SAVEA    DS     18F
BLANKS   DS     CL44' '
LOCLIST  CAMLST NAME,DSN,,CATBLOCK
DSN      DC     CL44' '
CATBLOCK DS     ØD
         DC     265X'ØØ'
         YREGS
         END
 $DB2TRM:         /* GENERAL DB2 ERROR REPORTING ROUTINE          */
 PROC (SQLCA,     /* INPUT - SQL COMMUNICATION AREA               */
       MTO,       /* INPUT - MTO PCB FOR TP PROGRAMMES ONLY       */
       CALLID)    /* INPUT - CODE TO INDENTIFY SOURCE OF CALL     */
       REORDER ;
 $DB2RRC:         /* ENTRY POINT FOR COBOL CALLERS                */
 ENTRY (SQLCA,    /* PARAMETERS MAP AS ABOVE                      */
       MTO,
       CALLID)
       OPTIONS(COBOL) REORDER;
```

$DB2TRM

```
-/*****************************************************************/
 /*           MODULE NAME: $DB2TRM                                */
 /*                                CURRENT VERSION# : Ø1          */
 /*                                                               */
 /* FUNCTION: $DB2TRM WILL REPORT UNEXPECTED DB2 ERRORS IN        */
 /*      BATCH PROGRAMS.                                          */
 /*          $DB2TRM WILL CAUSE USING PROGRAM TO TERMINATE        */
 /*           ABNORMALLY. THIS RESULTS IN THE FOLLOWING :         */
 /*             - PROGRAM'S MESSAGE OUTPUT TO THE USER TERMINAL   */
 /*               WILL NOT BE SENT. THIS ROUTINE SHOULD THEREFORE*/
```

```
 /*               BE USED ONLY IN EXTREME CASES THAT CANNOT BE   */
 /*               HANDLED BY THE PROGRAM.                        */
 /*             - IF THE PROGRAM IS AN UPDATE, ALL UPDATES ARE   */
 /*               BACKED OUT TO THE PREVIOUS SYNCH POINT.        */
 /*                                                              */
 /* ASSUMPTIONS:(1) PCB SUPPLIED FOR MASTER TERMINAL MUST HAVE   */
 /*                  "EXPRESS=YES"  (FOR USE IN TP PROGRAMME)    */
 /*             (2) $DB2TRM DOES NOT RETURN TO CALLING ROUTINE.  */
 /*             (3) THIS IS A BATCH TSO PGM.                     */
 /*--------------------------------------------------------------*/
 /* PARAMETER LIST :                                             */
 /* NO.  USAGE   NAME        DESCRIPTION        ATTRIBUTES       */
 /* === ====== ======= ====================   ================== */
 /*  1. INPUT  SQLCA   SQL COMMUNICATION          STRUCTURE      */
 /*                    AREA.                                     */
 /*  2. INPUT  MTO     (1) FOR USE IN BATCH       POINTER        */
 /*                     AND BMP, SET PTR=NULL                    */
 /*                    (2) FOR USE IN TP,                        */
 /*                       POINTER TO MTO PCB                     */
 /*                       WITH EXPRESS=YES                       */
 /*  3. INPUT  CALLID  CALL IDENTIFIER            CHAR(7)        */
 /*                     "MMMM-NN" WHERE                          */
 /*                      MMMM=MODULE NAME CALLING                */
 /*                        NN=CALL # IN MODULE                   */
 /*--------------------------------------------------------------*/
 /* MODULES CALLED BY THIS MODULE :                              */
 /* MODULE NAME  INT/EXT           FUNCTION DESCRIPTION          */
 /* ===========  ========  ======================================= */
 /* $PLITRM      EXT       ABNORMAL TERMINATION OF PL/I PGM      */
 /****************************************************************/
Ø/****************************************************************/
 /*                     C H A N G E   L O G                      */
 /*--------------------------------------------------------------*/
 /* PROGRAMMER |  DATE  | VSN |     DESCRIPTION OF CHANGE        */
 /*--------------------------------------------------------------*/
 /*$SUN        |90/09/18| Ø1  |MODULE WAS CREATED TO PRODUCE     */
 /*            |        |     |A DETAILED ERROR MESSAGE FOR ANY  */
 /*            |        |     |SQL STATEMENT THAT PRODUCED A     */
 /*            |        |     |RETURN CODE THAT REQUIRES THE     */
 /*            |        |     |TRANSACTION TO BE BROUGHT DOWN.   */
 /*--------------------------------------------------------------*/
 /*IMVN        |ØØ/Ø5/3Ø| Ø2  |ADDED ENTRY POINT FOR COBOL CALLER*/
 /****************************************************************/
1/*             PARAMETER DECLARATIONS                             */
 /****************************************************************/
 /*                                                              */
 /*       SQL COMMUNICATION AREA                                 */
 /*                                                              */
 /****************************************************************/
```

```
 DECLARE
   1 SQLCA     CHAR(136);
ØDCL MTO       POINTER;     /* MASTER TERMINAL PCB POINTER       */
ØDCL CALLID    CHAR(7);     /* CALL ID FOR MODULE POSITION       */
-DCL $PLITRM ENTRY(FIXED BIN(31));
-DCL PLIRETV     BUILTIN ;
 DCL Ø1 MSG,               /* OUTPUT HEADER.                    */
   Ø3 LL FIXED BIN(31) INIT(83),
   Ø3 Z1 BIT(8) ALIGNED INIT('ØØØØØØØØ'B),
   Ø3 Z2 BIT(8) ALIGNED INIT('Ø1ØØØØØØ'B),
   Ø3 DB2ERR CHAR(72)
           INIT(((27)'*'||' $DB2ERR MESSAGE '||(27)'*')),
   Ø3 Z3 CHAR(7) INIT(' ');
1DCL Ø1 MSG1,
       Ø3 LL FIXED BIN(31) INIT(83),
       Ø3 Z1 BIT(8) ALIGNED INIT('ØØØØØØØØ'B),
       Ø3 Z2 BIT(8) ALIGNED INIT('Ø1ØØØØØØ'B),
       Ø3 MSGEXP   CHAR(79);
-DCL Ø1 MSG2,
       Ø3 LL FIXED BIN(31) INIT(83),
       Ø3 Z1 BIT(8) ALIGNED INIT('ØØØØØØØØ'B),
       Ø3 Z2 BIT(8) ALIGNED INIT('ØØØØØØØØ'B),
       Ø3 MSGEXP   CHAR(79);
 DCL  ARG3   FIXED BIN(31) STATIC INIT(3);
 DCL  ARG4   FIXED BIN(31) STATIC INIT(4);
 DCL  ISRT   CHAR(4)       STATIC INIT('ISRT');
 DCL  I      FIXED BIN(15);
-DCL Ø1 PCB_MTO BASED(MTO),
        Ø2 LTERM CHAR(8),         /* LOGICAL TERMINAL NAME */
        Ø2 RSVD CHAR(2),          /* RESERVED FOR IMS */
        Ø2 STATUS CHAR(2),        /* RETURN STATUS CODE */
        Ø2 DATX FIXED DEC(7),     /* ØØYYDDD DATE RECEIVED BY IMS */
        Ø2 TIMX FIXED DEC(7),     /* HHMMSS.S TIME RECEIVED BY IMS */
        Ø2 MSG_SEQ FIXED BIN(31), /* MESSAGE SEQUENCE NUMBER TODAY */
        Ø2 MOD_NM CHAR(8);        /* PREDEFINED OUTPUT MOD NAME */
-DCL DFMOD CHAR(8) INIT('DFSMO2');   /* DEFAULT MOD NAME         */
1 /*****************************************************************/
  /*                                                               */
  /*                 SQL ERROR CODE HANDLING                       */
  /*                                                               */
  /*****************************************************************/
   DCL DSNTIAR                    ENTRY OPTIONS(ASM,INTER,RETCODE) ;
   DCL DATA_LEN                   FIXED BIN(31) INIT(8Ø) ;
   DCL DATA_DIM                   FIXED BIN(31) INIT(12) ;
   DCL 1 DB2_ERROR_MESSAGE        AUTOMATIC,
       2 ERROR_LEN                FIXED BIN(15) UNAL
                                 INIT((DATA_LEN * DATA_DIM)) ,
       2 ERROR_TEXT (DATA_DIM)    CHAR(DATA_LEN) ;
   DCL DSNTIAR_MSG                CHAR(58) INIT(
```

```
      'NON-ZERO RETURN CODE FROM CALL TO DSNTIAR, RETURN CODE IS') ;
-DCL DATE        BUILTIN;
 DCL HBOUND      BUILTIN;
 DCL NULL        BUILTIN;
 DCL SUBSTR      BUILTIN;
 DCL TIME        BUILTIN;
1 /*****************************************************************/
  /*                                                               */
  /*                 MAINLINE                                      */
  /*                                                               */
  /*****************************************************************/
     CALL FIND;
     CALL POST;
     CALL $PLITRM(3000);
1 /*****************************************************************/
  /*                                                               */
  /*              FIND THE SQL ERROR MESSAGE                       */
  /*                                                               */
  /*****************************************************************/
- FIND: PROC ;            /* FIND MESSAGE EXPANSION  */
    CALL DSNTIAR (SQLCA,DB2_ERROR_MESSAGE,DATA_LEN) ;
    IF PLIRETV ¬= 0 THEN DO ;
      PUT SKIP EDIT (DSNTIAR_MSG,PLIRETV)(A) ;
      PUT SKIP EDIT (DSNTIAR_MSG,PLIRETV)(A) ;
      PUT SKIP EDIT (DSNTIAR_MSG,PLIRETV)(A) ;
    END;  /* DO */
  END;  /* END FIND */
1 /*****************************************************************/
  /*                                                               */
  /*              WRITE THE MESSAGE TO THE MTO                     */
  /*                                                               */
  /*****************************************************************/
   POST: PROC ;
      SUBSTR(DB2ERR,10,7) = CALLID ;
      SUBSTR(DB2ERR,47,6) = DATE() ;
      SUBSTR(DB2ERR,55,6) = SUBSTR(TIME(),1,6) ;
      SUBSTR(DB2ERR,63,5) = 'VSN02' ;  /* CHANGE FOR NEW VERSIONS*/
      PUT SKIP(3) EDIT(DB2ERR)(A);
      DO I=1 TO HBOUND(DB2_ERROR_MESSAGE.ERROR_TEXT,1) ;
        PUT SKIP EDIT(DB2_ERROR_MESSAGE.ERROR_TEXT(I))(A);
      END;
      PUT SKIP EDIT(DB2ERR)(A);
   END;
 END;     /* END OF PROCEDURE                                    */
```

Michael Vanner
Senior Data Base Administrator
Canadian Tire Corp (Canada)

CHANGELIMIT image copy information – part 2

This month we conclude the code to present image copy information on screen.

ICOPYP1 – PANEL

```
)Attr Default(%+_)
   | type(text)   intens(high) caps(on ) color(yellow)
   ? type(text)   intens(high) caps(on ) color(green) hilite(reverse)
   # type(text)   intens(high) caps(off) hilite(reverse)
   ¬ type(output) intens(low ) caps(off) just(asis ) color(turquoise)
   } type(output) intens(low ) caps(off) just(asis ) color(yellow)
)Body  Expand(//)
%-/-/- ? List of tables +%-/-/-
%Command ===>_zcmd                                      / /%Scroll
===>_amt +
+Tablespace:}dbts            +
+-------------------------------------------------------------------------
#Table             #Creator #      Card#Last runstat       +
)Model
¬z                 ¬z       ¬z         }z                  +
)Init
  .ZVARS = '(tbname creator card stime)'
  &amt = CSR
)Reinit
)Proc
)End
```

ICOPYP2 – PANEL

```
)Attr Default(%+_)
   | type(text)   intens(high) caps(on ) color(yellow) hilite(reverse)
   [ type(text)   intens(high) caps(on ) color(red)    hilite(reverse)
   ( type(text)   intens(high) caps(on ) color(blue)   hilite(reverse)
   ? type(text)   intens(high) caps(on ) color(green)  hilite(reverse)
   { type(text)   intens(high) caps(on ) color(green)
   # type(text)   intens(high) caps(off) hilite(reverse)
   ¬ type(output) intens(low ) caps(off) just(asis ) color(white)
   } type(output) intens(low ) caps(off) just(asis ) color(yellow)
   ] type(output) intens(low ) caps(off) just(asis ) color(red)
   ) type(output) intens(low ) caps(off) just(asis ) color(blue)
   _ type(input)  intens(low ) caps(off) just(right) color(red)
)Body  Expand(//)
%-/-/- ? Image Copy Changelimit Report +%-/-/-
```

```
%Command ===>_zcmd                                        / /%Scroll
===>_amt +
+-------------------------------------------------------------------------
{Snapshot date:¬hdate     {time:¬htime   +                {>_pct  +
+-------------------------------------------------------------------------
                      #       4KB#     Empty#   Changed[Percent of+
(DB.TS           #Part#     Pages#     Pages#     Pages[ Changed  |
Ictype   +
)Model
)z               ¬z  ¬z         ¬z         ¬z         ]z         }z
+
)Init
  .ZVARS = '(dbts num kpag epag cpag ppag ict)'
  &amt = CSR
)Reinit
)Proc
)End
```

ICOPYP3 – PANEL

```
)Attr Default(%+_)
   | type(text)   intens(high) caps(on ) color(yellow)
   $ type(output) intens(high) caps(off) color(yellow)
   ? type(text)   intens(high) caps(on ) color(green) hilite(reverse)
   # type(text)   intens(high) caps(off) hilite(reverse)
   } type(text)   intens(high) caps(off) color(yellow) hilite(reverse)
   [ type( input) intens(high) caps(on ) color(green) pad(_)
)Body  Expand(//)
|-/-/- ? Image Copy Changelimit Strategy +|-/-/-
%Command ===>_zcmd
+
+
#PARAMETER  #PARAMETER VALUE                          #PROMPT
+
+
+SSID     =>[db2 +                              DB2 Sub-System Identifier
+Creator  =>[crec    +                          Table Creator
+Name     =>[tabc             +                 Table Name
+Tsname   =>[tsnc    +                          Tablespace Name
+Dbname   =>[dbnc    +                          Database   Name
+LowLimit =>[llim+                              Procent value Low  Limit
+HighLimit=>[hlim+                              Procent value High Limit
+
            $msg
+
+
                                } PF3 Return +
)Init
  &msg = 'Enter parameter values for the Changelimit Strategy !'
```

```
  if (&db2 ¬= ' ')
      .attr (db2) = 'pad(nulls)'
  if (&crec ¬= ' ')
      .attr (crec) = 'pad(nulls)'
  if (&tabc ¬= ' ')
      .attr (tabc) = 'pad(nulls)'
  if (&tsnc ¬= ' ')
      .attr (tsnc) = 'pad(nulls)'
  if (&dbnc ¬= ' ')
      .attr (dbnc) = 'pad(nulls)'
  if (&llim ¬= ' ')
      .attr (llim) = 'pad(nulls)'
  if (&hlim ¬= ' ')
      .attr (hlim) = 'pad(nulls)'
)Reinit
)Proc
  VPUT (db2 crec tabc tsnc dbnc llim hlim) PROFILE
)End
```

ICOPYP4 – PANEL

```
)Attr Default(%+_)
   ( type(text  ) intens(high) hilite(reverse)
   ] type(text  ) intens(high) hilite(reverse) color(green)
   [ type(text  ) intens(high) hilite(reverse) color(yellow)
   ~ type(output) intens(high) color(red)
   ? type(text)   intens(high) caps(on ) color(green) hilite(reverse)
   + type(text  ) intens(low )
   _ type( input) intens(high) caps(on ) just(left)
   ¬ type(output) intens(low ) caps(off) just(asis)
)Body  Expand(//)
%-/-/- ? Selection Result +%-/-/-
+
+Command ===>_zcmd                                          +Scroll
===>_amt +
+
+Press]Enter+to have this service continue.
+Press]End  +to  respecify your PARAMETERS.
+
]Dbname  +]Tsname  +
)Model
¬z       +¬z       +
)Init
  .ZVARS = '(db ts )'
  &amt = PAGE
)Reinit
)Proc
)End
```

PICOPY – PL/I SOURCE CODE

```
* PROCESS GS,OFFSET,OPT(TIME);
 PICOPY:PROC(PARMS)OPTIONS(MAIN) REORDER;
 /*****************************************************************/
 /* DESCRIPTION: IMAGE COPY INFORMATION                           */
 /*****************************************************************/
  DCL PARMS CHAR(1ØØ) VAR;
  DCL SYSPRINT    FILE STREAM OUTPUT;
  DCL NUMSEQ        BIN FIXED(31) INIT(Ø);
  DCL (DB,TS)       CHAR(8) VAR;
  DCL (HDB,HTS)     CHAR(8);
  DCL HDATE         CHAR(26);
  DCL STATUS        CHAR(28);
  DCL DD            CHAR(4);
  DCL (HPART,HDAY)  BIN FIXED(15);
  DCL HDAY1         PIC'ZZZ9';
  DCL (SUBSTR,DATE,TIME,NULL,ADDR,LENGTH,INDEX) BUILTIN;
  DCL OUT           CHAR(8) VAR;
  DCL IC            BIN FIXED(15);
  EXEC SQL INCLUDE SQLCA;
  IF SUBSTR(PARMS,1,8)='%' THEN DB='%';
  ELSE DO;
     CALL FUNC(SUBSTR(PARMS,1,8),OUT);
     DB=OUT;
     IF LENGTH(DB) < 8 THEN DB=DB||'%';
  END;
  IF SUBSTR(PARMS,9,8)='%' THEN TS='%';
  ELSE DO;
     CALL FUNC(SUBSTR(PARMS,9,8),OUT);
     TS=OUT;
     IF LENGTH(TS) < 8 THEN TS=TS||'%';
  END;
  PUT SKIP LIST ('IMAGE COPY INFORMATION');
  /* NO IMAGE COPY                                              */
  EXEC SQL DECLARE C2 CURSOR WITH HOLD FOR SELECT
  DBNAME,TSNAME,PARTITION
  FROM SYSIBM.SYSTABLEPART S
  WHERE S.DBNAME LIKE :DB
    AND S.TSNAME LIKE :TS
    AND NOT EXISTS (SELECT 1 FROM SYSIBM.SYSCOPY
                    WHERE ICTYPE IN ('F','I')
                      AND DBNAME = S.DBNAME
                      AND TSNAME = S.TSNAME)
  FOR FETCH ONLY;
  EXEC SQL OPEN C2;
  EXEC SQL FETCH C2 INTO :HDB, :HTS, :HPART;
  DO WHILE (SQLCODE=Ø);
     NUMSEQ=1;
     STATUS=' NO IMAGE COPY FOUND';
     PUT SKIP LIST
```

```
     (HDB||' '||HTS||' '||HPART||' - '||STATUS);
     EXEC SQL FETCH C2 INTO :HDB, :HTS, :HPART;
  END;
  EXEC SQL CLOSE C2;
  EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR SELECT
  S.DBNAME,S.TSNAME,PARTITION,MAX(TIMESTAMP),
  MIN(DAYS(CURRENT TIMESTAMP)-DAYS(TIMESTAMP))
  FROM SYSIBM.SYSTABLEPART  S,
       SYSIBM.SYSCOPY C
  WHERE S.DBNAME LIKE :DB
    AND S.TSNAME LIKE :TS
    AND ICTYPE IN ('F','I')
    AND C.DBNAME = S.DBNAME
    AND C.TSNAME = S.TSNAME
  GROUP BY S.DBNAME, S.TSNAME, PARTITION
  ORDER BY 5 DESC
  FOR FETCH ONLY;
  EXEC SQL OPEN C1;
  EXEC SQL FETCH C1 INTO :HDB, :HTS, :HPART, :HDATE, :HDAY;
  DO WHILE (SQLCODE=Ø);
     NUMSEQ=1;
     IF HDAY=1
     THEN DD='DAY';
     ELSE DD='DAYS';
     HDAY1=HDAY;
     STATUS=' '||HDAY1||' '||DD||' OLD';
     IF HDAY=Ø
     THEN STATUS=' TODAY IMAGE COPY';
     PUT SKIP LIST
     (HDB||' '||HTS||' '||HPART||' '||SUBSTR(HDATE,1,19)||STATUS);
     EXEC SQL FETCH C1 INTO :HDB, :HTS, :HPART, :HDATE, :HDAY;
  END;
  EXEC SQL CLOSE C1;
  IF NUMSEQ=Ø THEN PUT SKIP LIST ('NO CATALOG ENTRIES FOUND');
  FUNC:PROC(INP,OUT);
       DCL INP CHAR(8);
       DCL OUT CHAR(8) VAR;
       DO IC=1 TO 8 BY 1 WHILE (SUBSTR(INP,IC,1) ¬=' ');
       END;
       OUT=SUBSTR(INP,1,IC-1);
   END FUNC;
 END PICOPY;
```

PICOPY1 – PL/I SOURCE CODE

```
* PROCESS GS,OFFSET,OPT(TIME);
 PICOPY1:PROC(PARMS)OPTIONS(MAIN) REORDER;
 /*****************************************************************/
 /* DESCRIPTION: LIST OF TABLES                                   */
 /*****************************************************************/
```

```
 DCL PARMS CHAR(100) VAR;
 DCL SYSPRINT    FILE STREAM OUTPUT;
 DCL HCARD         BIN FIXED(31);
 DCL MCARD         PIC'--.---.---.--9';
 DCL (DB,TS)       CHAR(8);
 DCL HTB           CHAR(18);
 DCL HCR           CHAR(8);
 DCL HDATE         CHAR(26);
 DCL (SUBSTR,DATE,TIME,NULL,ADDR,LENGTH,INDEX) BUILTIN;
 EXEC SQL INCLUDE SQLCA;
 DB=SUBSTR(PARMS,1,8);
 TS=SUBSTR(PARMS,9,8);
 PUT SKIP LIST('LIST OF TABLES');
 EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR SELECT
 CREATOR, NAME, CARD, STATSTIME
 FROM SYSIBM.SYSTABLES
 WHERE TSNAME = :TS
   AND DBNAME = :DB
   AND TYPE = 'T'
 ORDER BY CREATOR,NAME
 FOR FETCH ONLY;
 EXEC SQL OPEN C1;
 EXEC SQL FETCH C1 INTO :HCR, :HTB, :HCARD, :HDATE;
 DO WHILE (SQLCODE=0);
    MCARD=HCARD;
    PUT SKIP LIST
    (HTB||' '||HCR||' '||MCARD||' '||SUBSTR(HDATE,1,19));
    EXEC SQL FETCH C1 INTO :HCR, :HTB, :HCARD, :HDATE;
 END;
 EXEC SQL CLOSE C1;
END PICOPY1;
```

PICOPY2 – PL/I SOURCE CODE

```
* PROCESS GS,OFFSET,OPT(TIME);
 PICOPY2:PROC(PARMS)OPTIONS(MAIN) REORDER;
 /*****************************************************************/
 /* DESCRIPTION: SELECTION RESULT                                 */
 /*****************************************************************/
  DCL PARMS CHAR(100) VAR;
  DCL SYSPRINT    FILE STREAM OUTPUT;
  EXEC SQL INCLUDE SQLCA;
  /* INPUT DECLARATIONS                          */
  DCL 1 WORKST,
      2 CRE         CHAR(8)  VAR,
      2 TAB         CHAR(18) VAR,
      2 TSN         CHAR(8)  VAR,
      2 DBN         CHAR(8)  VAR;
  /* HOST VARIABLES FOR SYSIBM.SYSTABLES        */
  DCL HDATE        CHAR(10);
```

```
DCL HTIME        CHAR(8);
DCL DBNAME       CHAR(8);
DCL TSNAME       CHAR(8);
DCL IC            BIN FIXED(15);
DCL OUT           CHAR(18) VAR;
/* INIT INPUT VARIABLES                        */
IF SUBSTR(PARMS,1,8)=' ' THEN CRE='%';
ELSE DO;
   CALL FUNC(SUBSTR(PARMS,1,8),OUT);
   CRE=OUT;
   IF LENGTH(CRE) < 8 THEN CRE=CRE||'%';
END;
IF SUBSTR(PARMS,9,18)=' ' THEN TAB='%';
ELSE DO;
   CALL FUNC(SUBSTR(PARMS,9,18),OUT);
   TAB=OUT;
   IF LENGTH(TAB) < 18 THEN TAB=TAB||'%';
END;
IF SUBSTR(PARMS,27,8)=' ' THEN TSN='%';
ELSE DO;
   CALL FUNC(SUBSTR(PARMS,27,8),OUT);
   TSN=OUT;
   IF LENGTH(TSN) < 8 THEN TSN=TSN||'%';
END;
IF SUBSTR(PARMS,35,8)=' ' THEN DBN='%';
ELSE DO;
   CALL FUNC(SUBSTR(PARMS,35,8),OUT);
   DBN=OUT;
   IF LENGTH(DBN) < 8 THEN DBN=DBN||'%';
END;
/* CHECK, IF TABLE(S) EXISTS                    */
EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR SELECT
DISTINCT DBNAME, TSNAME
FROM SYSIBM.SYSTABLES
WHERE CREATOR LIKE :CRE
  AND NAME    LIKE :TAB
  AND TSNAME  LIKE :TSN
  AND DBNAME  LIKE :DBN
  AND TYPE = 'T'
ORDER BY DBNAME,TSNAME
FOR FETCH ONLY;
EXEC SQL OPEN C1;
EXEC SQL FETCH C1 INTO :DBNAME, :TSNAME;
IF SQLCODE ¬= Ø THEN PUT SKIP LIST ('SQLCODE ='||SQLCODE);
EXEC SQL SELECT CURRENT DATE, CURRENT TIME
     INTO :HDATE, :HTIME
     FROM SYSIBM.SYSDUMMY1;
PUT SKIP LIST (' '||HDATE||' '||HTIME);
DO WHILE (SQLCODE=Ø);
   PUT SKIP LIST ('D '||DBNAME||' '||TSNAME);
   EXEC SQL FETCH C1 INTO :DBNAME, :TSNAME;
```

```
  END;
  EXEC SQL CLOSE C1;
  /* CONVERSION TO VARING VARIABLE                      */
  FUNC:PROC(INP,OUT);
       DCL INP CHAR(18);
       DCL OUT CHAR(18) VAR;
       DO IC=1 TO 18 BY 1 WHILE (SUBSTR(INP,IC,1) ¬=' ');
       END;
       OUT=SUBSTR(INP,1,IC-1);
   END FUNC;
 END PICOPY2;
```

ICCIS – JCL SKELETON

```
)TBA 72
)CM ---------------------------------------------------------------------
)CM Skeleton: JCL for Changelimit Information                          --
)CM ---------------------------------------------------------------------
//&user.X JOB (12ØØ-12Ø5-ØØ),'&option',
//             NOTIFY=&user,REGION=4M,
//             CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//* ******************************************************************
//*    &title
//*    GENERATION DATE AND TIME : &date AT: &time
//*
//*    ICCIS  - WAS RUN WITH THE FOLLOWING PARAMETERS:
//*    PARAMETER   PARAMETER VALUE
//*    ----------  ----------------------
//*    SSID     :  &db2
//*    Creator  :  &crec
//*    Name     :  &tabc
//*    Tsname   :  &tsnc
//*    Dbname   :  &dbnc
//*    LowLimit :  &llim
//*    HighLimit:  &hlim
//* ******************************************************************
//*
//*---- DELETE OLD DATASET ----------------------
//DELETE   EXEC PGM=IDCAMS,COND=(4,LT)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DELETE '&user..TEMP.DATA'
  SET MAXCC = Ø
/*
//*
//*---- COPY ----------------------------------------
//COPYS EXEC DSNUPROC,SYSTEM=&db2,
//      UID='&user..COPY',UTPROC='',COND=(4,LT)
//STEPLIB  DD DSN=DSN51Ø.SDSNLOAD,DISP=SHR
//SYSCOPY  DD DUMMY
```

```
//SYSPRINT DD UNIT=339Ø,
//         DSN=&user..TEMP.DATA,
//         DISP=(NEW,CATLG,CATLG),
//         DCB=(DSORG=PS,RECFM=FB,LRECL=131,BLKSIZE=131Ø),
//         SPACE=(CYL,(1,1),RLSE)
//SYSIN  DD  *
)DOT "ICLIST"
  COPY  TABLESPACE &DB..&TS CHANGELIMIT(&llim,&hlim) REPORTONLY
)ENDDOT
/*
```

ICI00 – MESSAGE

```
ICIØØ1           .ALARM = YES  .WINDOW=NORESP .ALARM = YES
'&message'
```

Bernard Zver (Slovenia)

Making sure that Business Intelligence aids intelligent business with DB2 Query Patroller

Business intelligence is not business as usual. It's about making better decisions and making them more quickly. Businesses collect enormous amounts of data every day – information about orders, inventory, accounts payable, point-of-sale transactions through point redemption programs, and, of course, customers. To add to the mountains of information, businesses also acquire data, such as demographics and mailing lists, from outside sources.

Consolidating and organizing this data for better business decisions can lead to a competitive advantage, and learning to uncover and leverage those advantages is what business intelligence is all about.

In the past, end users typically called on their IS department and asked for a report on the market, sales, or inventory. IS generated the report on a user's behalf and provided it usually days or even weeks later. These reports were static in nature and did not allow an end user to change the attributes of the report in order to gain a better insight into the information that was being analysed. Sometimes, requirements

were not communicated properly and the users waited weeks for a report that did not answer their business questions.

In the late 1980s, desktop query and reporting tools entered the market, allowing end users to perform their own queries against the corporate or departmental data warehouse or data marts. This immediately provided the companies using these applications with a competitive advantage. While end users at other companies waited weeks for a report in order to make a decision, companies that allowed their business analysts and key managers to query directly against the database were making decisions in minutes and gaining a competitive advantage in the marketplace. There was a push to extend this type of access to information to all users, so that each and every employee of a company had the power to make better, more informed, timelier decisions.

The steady increase in end users performing queries against the corporate data warehouse or data marts presents a huge challenge for database administrators. As the number increases, the response time of the system may decline because of increased contention. Large-scale data warehouses that provide breakthrough business value pose a challenge: how can a company's data warehouse continue to provide quick response time across large amounts of data to an ever-increasing number of end users tapping the power of *ad hoc* query tools on their desktops?

One solution to this problem has been to add more hardware. IBM's DB2 Universal Database is one of the most scalable relational database management systems in the industry, with DB2 Enterprise – Extended Edition offering near-linear scalability with additional database partition servers. Hardware and software scalability can, in part, help handle the increased load or help spread the load over several machines to improve query performance. This is the direction many leading edge organizations have taken with their Decision Support Systems (DSS).

However, even with the addition of new, more-powerful, hardware systems, users may still unknowingly submit 'runaway queries'. Runaway queries can be defined as those queries that are poorly constructed and consume vast amounts of computing resource. Their

impact on a DSS often comes as a total surprise to the user who submits them. For example, you may have a rather inexperienced user submit a query that requests a join of two tables without any predicates. The result set for this type of join is referred to as the Cartesian Product of the two tables – each row in table A is joined to each row in table B. As a simple example, try joining the STAFF (35 rows) and the ORG (8 rows) tables in DB2's SAMPLE database with the following SQL statement: db2 "select * from staff,org". The result? 280 rows!

With the advancement in query tools, it is now possible for every business analyst in your company to quickly generate a query without knowing anything about the back-end database or about SQL. When too many end users submit complex runaway queries to a DSS at the same time, they can potentially bring a large multi-terabyte system to its knees

Another example of end user mismanagement that leads to an increase in query response time is the resubmission of a query that is taking a long time to process. For example, assume that an end user unknowingly submits a complex query that could be classified as a runaway query. After the end user returns from lunch some hours after submitting the query, he or she notices that the result set has not been returned. The end user assumes that there was a system crash, and reboots his or her machine. When the machine restarts, the end user resubmits the query. The database server now has to contend with two runaway queries, slowing the performance of the DSS even more!

The rise in response time is due primarily to poor query management. If the query submissions had been controlled, for example by rejecting a runaway query based on its estimated resource cost, the DSS response time would not have been significantly affected. But because all the queries reached the data warehouse at the same time, and some may have been badly formulated or even worse submitted twice, the database load management capability was overwhelmed.

All of these trends point to the need to grow and protect the data warehouse as the vital corporate asset that it is. Since large data warehouses are enabling radical business breakthroughs while maintaining competitive advantages in the marketplace, their

robustness, availability, performance, and security are of paramount importance.

Today, database management personnel are facing increasing challenges. While they want to deliver this valuable information to end users as quickly as possible, they are finding that it takes an enormous amount of resource to be responsive to the growing number of users demanding the information. Even worse, the users' need for information continues to change and evolve, as new information becomes available. What does this lead to? A bottleneck to that 'time is money' information that leads to better more-informed decisions.

DB2 QUERY PATROLLER DEVELOPED TO MANAGE YOUR DSS

DB2 Query Patroller greatly improves the scalability of a data warehouse by allowing hundreds of users to safely submit queries on multi-terabyte systems. The product is a true three-tier architecture solution. Its components span the distributed environment to better manage and control all aspects of query submission.

DB2 Query Patroller acts as an agent on behalf of the end user. It prioritizes and schedules queries so that query completion is more predictable and computer resources are efficiently utilized. After an end user submits a query, DB2 Query Patroller frees up for other work or additional queries while waiting for the original query results. DB2 Query Patroller is integrated with the DB2 optimizer. It performs cost analysis on queries and then schedules and dispatches those queries so that the load is balanced across the database partitions.

DB2 Query Patroller sets individual user and group priorities, as well as user query limits. This enables the data warehouse to deliver the needed results to its most important users as quickly as possible. It also has the ability to limit usage of the system by stopping those runaway queries before they even start. If desired, an end user can choose to receive e-mail notification of scheduled query completion or query failure.

ARCHITECTURE OVERVIEW

DB2 Query Patroller consists of components running on the database

server and end users' desktops. DB2 Query Patroller is made up of several components each having a specific task in providing query and resource management.

DB2 Query Patroller Server

The DB2 Query Patroller Server is the core component of DB2 Query Patroller. It provides an environment for storing user profiles, setting up system parameters, maintaining job lists, and storing node information. A DB2 Query Patroller system administrator has a Java-based management tool that can be used to perform these tasks. The server component executes on a node with the DB2 server.

DB2 Query Patroller Agent

The agent component of DB2 Query Patroller resides on each database partition server. It processes the database requests on behalf of the DB2 Query Patroller Server and gathers resource utilization statistics to allow for query workload balancing as well as monitoring of the resource utilization of each partition.

In a DB2 Enterprise Edition environment, the agent and server components run on the same machine. In a DB2 Enterprise – Extended Edition environment, the server and an agent must run on one database partition server, with agents running on all other database partition servers that participate in the partitioned database.

Query Administrator

The Administrator tool gives a DBA or system administrator the ability to manage the DB2 Query Patroller environment. This Java-based tool allows for the management of the Query Patroller system. The Query Administrator provides menus to configure user profiles, system parameters, and node parameters allowing the system administrator to set up system-wide, partition-level or user-level thresholds for governing the data warehouse, including:

- Maximum number of queries running on the system at any given time.
- Maximum cost threshold for the entire system. The cumulative cost of all queries running cannot exceed this number.

- Maximum cost threshold for each defined user or group in the system.
- Maximum number of jobs a user can initiate. This value can be configured differently for each of your users or groups.
- Specific amount of time to retain temporary result tables. When DB2 Query Patroller takes control of a query, a temporary table is created in the database to store the query results. DB2 Query Patroller will automatically clean up these tables after the period of time specified by the administrator. Query Patroller will also allow users to share result sets so that a query can be executed once and all authorized users can reuse the result set.

Query Monitor

The Query Monitor tool of DB2 Query Patroller provides both the administrators and the end users with a Java-based interface for viewing and managing their queries. The Query Monitor component enables end users to view a job's status, submit and cancel queries, and drop result tables. Unlike system administrators, end users can only display information for their own queries and jobs running on their own system while the Query Monitor tool provides administrators with the ability to manage all queries submitted against the DSS.

An example of the Query Monitor tool is shown in Figure 1.

The DB2 Query Monitor job list maintains the queries submitted by end users. The job list contains information about each query submitted through Query Patroller. The system administrator or end user is able to use the job list to view information for the queries in the system including:

- Job sequence number
- Query priority
- Query status
- Query source
- Node on which the query was submitted

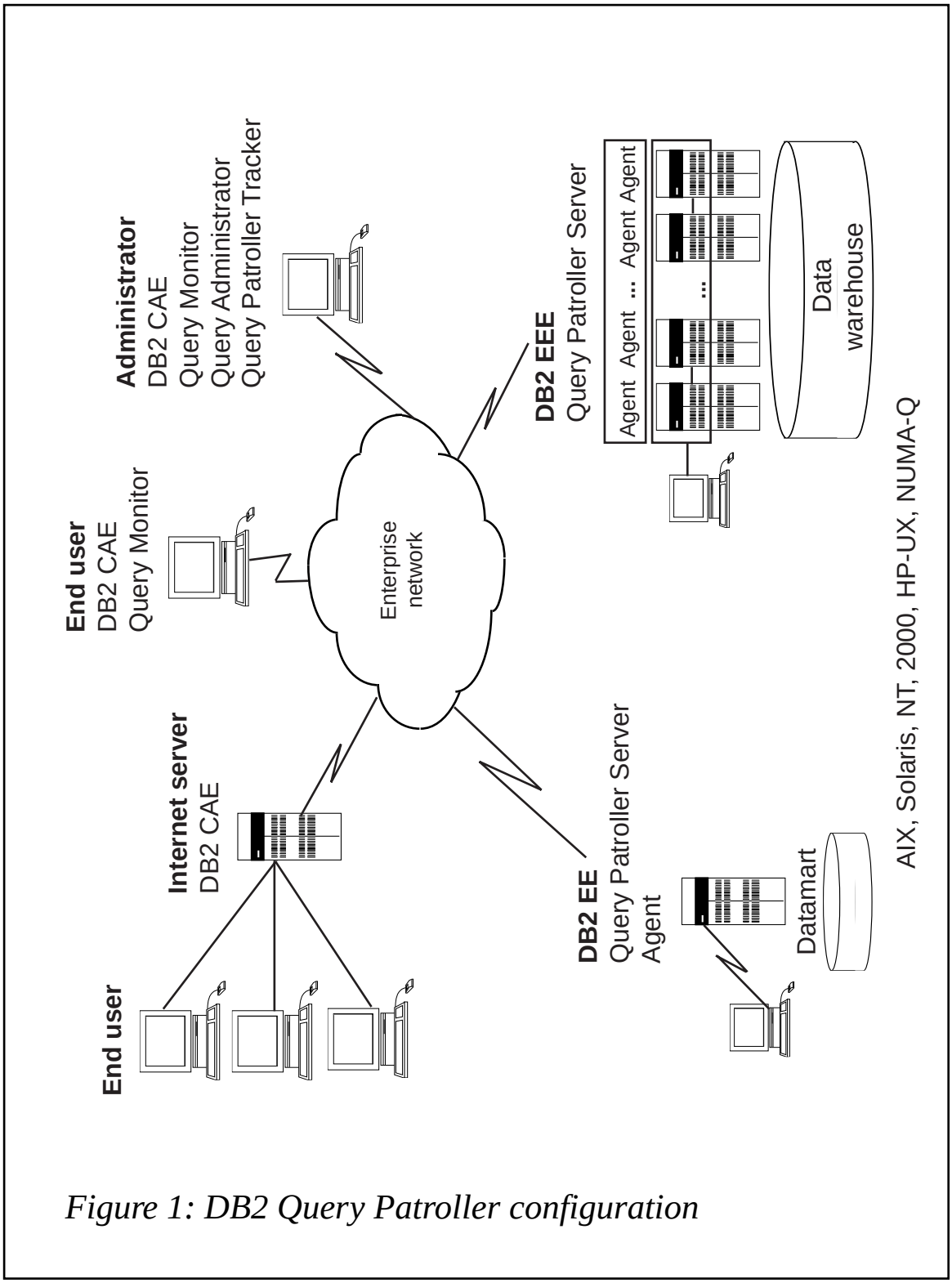

Figure 1: DB2 Query Patroller configuration

- Type of application submitting the query
- ID of user submitting the query

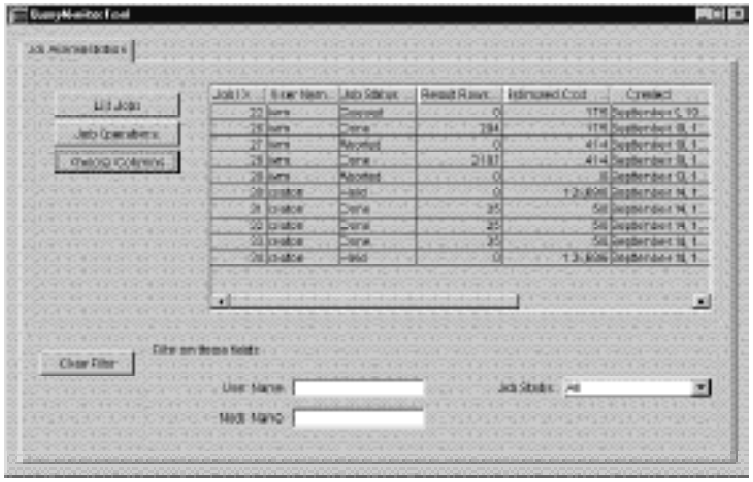

Figure 2: DB2 Query Monitor

- Date and time the query was submitted.

Users and administrators may also view more detailed information on any of the queries listed in the job list table, such as query run time, cost of the query, and the SQL statement.

Figure 2 gives a typical administrator's view of this tool. You can see that more than one user is shown. The administrator can also manipulate the granularity of what is shown in this window using the attributes in the 'Filter on these Fields' box.

Query Enabler

The Query Enabler tool of DB2 Query Patroller executes inside a DB2 client. This tool captures dynamic SQL statements being passed to DB2 from any front end query tool. This is an added enhancement to this product for Version 7. In DB2 Universal Database Version 6, you could only trap and monitor ODBC-based queries with DB2 Query Patroller.

Query Enabler interacts with other DB2 Query Patroller components, and with the user, to execute or schedule the query and to return results from previously completed queries.

Query Enabler intercepts queries submitted by end users. If matching queries exist on the Query Patroller Server, Query Enabler provides the end user with a display of those queries and prompts the user to indicate whether or not a new result set should be returned. Whenever a user wants to submit a query, Query Enabler provides the option to set scheduled run times or to submit and wait for the results. If the end user does not want to wait for the query results, Query Enabler releases the desktop application and passes the query to the DB2 Query Patroller Server. Query Patroller then takes control of the query and runs it in the background on behalf of the end user. The next time the user submits that same query, the result set for that query will be returned to the application.

Query Enabler also has the ability to run in a silent mode so that the end user does not interact with the Query Enabler, but rather submits queries directly to the server. To the user, the concept of Query Patroller and captured queries does not even exist, but to the administrator, it is a powerful hidden tool to control the usage of the corporate DSS. Query Patroller's silent mode also enables 3-tier or n-tier applications to utilize Query Patroller without the need for additional software on the client's desktop.

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

Paul Zikopoulos
Database Specialist with the DB2 Sales Support team
IBM Canada (Canada)

Call for papers

Why not share your expertise and earn money at the same time? *DB2 Update* is looking to swell the number of contributors who send in technical articles, hints and tips, and utility programs, etc. We would also be interested in articles about performance and tuning, and information and tips for DB2 DBAs. If you have an idea for an article, or you would like a copy of our *Notes for Contributors*, contact the editor, Trevor Eddolls, at any of the addresses shown on page 2.

November 1996 – October 2000 index

Items below are references to articles that have appeared in *DB2 Update* since November 1996. References show the issue number followed by the page number(s). Back-issues of *DB2 Update* are available back to Issue 15 (January 1994). See page 2 for details.

DB2 news

IBM has announced Version 7.1 of its DB2 Everyplace for linking and synchronizing mobile devices with central DB2 data and applications. It's also used for creating mobile applications for Palm devices by using the Personal Application Builder.

It combines development, deployment, and administration in one package, and is backed by the synchronization capabilities of the DB2 Everyplace Sync Server, allowing mobile and enterprise employees to view and update data simultaneously.

Supported mobile devices include Palm OS 3.0 or later, Windows CE, 2.0 or later, EPOC Release 5 or later, Linux and embedded Linux Kernel 2.2 or later, and Neutrino 2.0+.

Supported servers include Windows 2000 or NT with Service Pack 4 or later, DB2 Universal Database Enterprise Edition Version 7 for Windows, and IBM Websphere 3.02 or later.

A trial copy can be downloaded from IBM's Web site – http://www.ibm.com/software/data/db2.

For further information contact your local IBM representative.

* * *

Sand Technology has ported its Nucleus exploration data warehousing and business intelligence software to run on the AS/400.

It complements the AS/400's DB2 Universal Data Base and other data management and business intelligence packages from IBM and its partners. It already runs on 64-bit Unix and 32-bit Windows 9*x* and NT.

The software allows users to look at large amounts of data randomly and non-repetitively. It's built on token-based technology that allows data to be condensed to the point that it can be placed in memory, so analysis and retrieval speeds are faster.

For further information contact:
Sand Technology, 4141, Sherbrooke Street West, Suite 410, Westmount, Quebec, Montreal, Canada H3Z 1B8.
Tel: (514) 939 3477.
Sand Technology (UK) Ltd, Maple House, High Street, Potters Bar, Herts, EN6 5BS, UK.
Tel: (01462) 828005.
URL: http://www.sandtechnology.com/sandtechnology.html.

* * *

IBM has Version 7.1 of its Enterprise Information Portal, promising enhanced data access, expanded search capabilities, information mining, and integration of workflow processes across all data stores.

New capabilities include access to relational databases, including DB2, Oracle, and others that support ODBC or JDBC connections. This adds structured data support to the current connections for unstructured information such as PC documents, audio, video, and images.

EIP also provides access to non-relational datastores, including VSAM and IMS, via IBM DataJoiner.

For further information contact your local IBM representative.
URL: http://www-4.ibm.com/software/data/eip.