# 124

# DB2

*February 2003*

## In this issue



update

# DB2 Update


## Published by

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

## North American office

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

## Subscriptions and back-issues

A year's subscription to *DB2 Update*, comprising twelve monthly issues, costs £255.00 in the UK; $380.00 in the USA and Canada; £261.00 in Europe; £267.00 in Australasia and Japan; and £265.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1999 issue, are available separately to subscribers for £22.50 ($33.75) each including postage.

## *DB2 Update* on-line

Code from *DB2 Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/db2; you will need to supply a word from the printed issue.

## Editor

Trevor Eddolls


## Disclaimer

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

## Contributions

When Xephon is given copyright, articles published in *DB2 Update* are paid for at the rate of £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 download a copy of our *Notes for Contributors* from www.xephon.com/nfc.




*Printed in England.*

# Distributed processing on DB2 for OS/390 – things to consider

Distributed data processing on DB2 for OS/390 server is becoming very common in enterprises. I will present some information about defining the DB2 parameters associated with distributed processing and for performance monitoring. The information has been compiled from several sources. Though not exhaustive, it should give enough information to manage and administer database access on the host from a desktop client. This discussion assumes that all remote access to the database is through dynamic SQL using protocols like ODBC or JDBC.

Terminology and concepts:

- Distributed database access or remote database access is facilitated by the DDF (Distributed Data Facility) component of DB2, which will be seen as a started task named like *SSID*DIST, where *SSID* is the four-character DB2 subsystem identifier.
- Every time there is a remote access, it is executed under the plan DISTSERV. This plan is automatically created at execution time.
- A ‘server’ is the DB2 subsystem that is processing an SQL request from an application or client that is the ‘requester’.
- DRDA (Distributed Relational Database Architecture) is a protocol for accessing DB2 data on OS/390 and other hosts through the DDF.
- Database access threads (remote threads) are created to access data on a DB2 server on behalf of a requester using the DRDA protocol. Allied threads, on the other hand, are those that perform work at the local site where the request is made.
- An inactive thread may be defined as a database access thread that does not hold any cursors or database resources.

1 The DDF THREADS (CMTSTAT) parameter needs to be set to INACTIVE for the following reasons:

- A database access thread persists until the connection between the requester and the server terminates. The value INACTIVE for this parameter would make the thread inactive when the application requests have been served. When another request is made, the inactive thread is re-used and becomes active, saving the cost of a thread creation. For this reason, these threads are also referred to as 'sometimes active' threads. If the parameter is set to 'ACTIVE' then the thread will be kept active even after the application request has been served, but idle, and cannot be used by any other request. It will eventually get cancelled based on the IDLE THREAD TIMEOUT parameter.
- This would help achieve the maximum number of 25,000 DDF threads instead of the 2000 when the parameter is set to ACTIVE.
- Less storage is used for each DDF thread.
- You get an accounting trace record (IFCID 0003) each time a thread becomes inactive rather than once for the entire time you are connected. When an inactive thread becomes active, the accounting fields for that thread are initialized again. As a result, the accounting record contains information about active threads only. This makes it easier to study how distributed applications are performing.
- Each time a thread becomes inactive, workload manager resets the information it maintains on that thread. The next time that thread is activated, workload manager begins managing the goals you have set for transactions that run in that service class. If you use multiple performance periods, it is possible to favour short-

 

running units of work that use fewer resources while giving fewer resources over time to long-running units of work. The response times reported by RMF do not include inactive periods between requests.

- This makes it more practical to take advantage of the ability to time-out idle active threads.

2 The IDLE THREAD TIMEOUT (IDTHTOIN) parameter must be set to an appropriate value. Usually 300 seconds would be adequate. However, you must consider the fact that DB2 would poll for idle threads every three minutes and hence it would be possible for the threads to be idle for longer than 300 seconds. If using TCP/IP-based connectivity to the DB2 server, the KEEPALIVE timer for TCP/IP on the application requester must be set to the same as or lower than that of the server using the parameter KEEPALIVEOPTIONS with the keyword INTERVAL.

When the keep-alive timer detects a network failure and notifies DB2, message DSNL511I is displayed on the MVS system log and the DB2 master address space.

If the thread is cancelled because of the IDLE THREAD TIMEOUT parameter, message DSNL027I is displayed.

In both cases, the thread is cancelled, and the updates are rolled back.

3 The CACHE DYNAMIC SQL (CACHEDYN) parameter must be set to YES to enable the Dynamic SQL to be cached and reused. This is called global cacheing or prepared statement cacheing. Cacheing can also be done using the bind parameter KEEPDYNAMIC (YES) for the plan/package. These options are beneficial when the SQL needs to be prepared. Enabling prepared SQL statement cacheing will avoid an unnecessary prepare and will provide for faster response for repetitive SQL. The SQL may come from different sources, but they must be identical even to the smallest white space for cacheing to be effective. However, be aware that enabling prepared SQL statement cacheing

will require an increase in the EDM pool and, sometimes, it may be detrimental to the statically bound SQL also.

4 The TCP/IP ALREADY VERIFIED (TCPALVER) parameter determines whether TCP/IP connection requests containing only a userid (and no password, RACF PassTicket, or DCE ticket) are accepted by DB2. If YES is specified, TCP/IP connection requests containing only a userid are accepted. The default is NO. Because this has security implications, it will be prudent to set it to NO.

5 The Resource limit table access error (RLFERRD) parameter specifies the default action to take or the default limit to use for an *ad hoc* query from a remote location if the resource limit table has not been updated appropriately for database changes. The values are NORUN, NOLIMIT, and a value between 1 and 5,000,000 for the CPU seconds allowed for a remote query. NOLIMIT is preferred here, because it will allow any query to run.

6 The Resync interval (RESYNC) parameter indicates the number of minutes between resynchronization periods and can range between 0 and 99. A value between 2 and 5 is preferable.

The following are other non-DDF parameters to consider (these are installation-dependent and are significant only when there are acute performance issues):

- MAX REMOTE CONNECTED (CONDBAT) – between 0 and 25,000.
- MAX CONCURRENT USERS (CTHREAD) – between 1 and 2,000.
- MAX REMOTE ACTIVE (MAXDBAT) – between 1 and 1,999 and must be less than or equal to CONDBAT.

## PERFORMANCE MEASUREMENTS

Different performance indicators related to distributed processing

could be captured and evaluated. There are several IFCIDs under the statistics and accounting traces that have useful information related to distributed processing. Accounting information may be evaluated under the DISTSERV plan. If the DDF THREAD (CMTSTAT) parameter has been set to INACTIVE, a new accounting record will be written every time a thread is reused. Some of the trace information is listed below.

To track the use or effectiveness of cacheing of dynamic SQL, the following fields of IFCID 2 in the statistics record generated by Statistics trace class 1 will be useful. This is a low-overhead trace, which is generally turned on in the DB2 subsystem.

- QXSTFND – number of successful cache searches.
- QXSTNFND – number of unsuccessful cache searches.
- QXSTIPRP – number of implicit PREPAREs.
- QXSTNPRP – number of PREPAREs avoided following a commit point when KEEPDYNAMIC(YES) and cache are active.
- QXSTDEXP – number of times inactive prepared statements were discarded from DBM1 because the limit was exceeded.
- QXSTDINV – number of times a cached statement was purged from the cache because a DROP, ALTER, or REVOKE statement was executed.

To track the thread use for distributed processing, the following fields in IFCID 1 of the statistics trace class 1 will be useful:

- QDSTQDBT – number of times that a database access thread was queued because it reached the zparm maximum for active remote threads. If this value contains a large number you might want to increase the maximum number of database access threads (MAXDBAT) allowed.
- QDSTQCRT – number of conversations that were deallocated because the zparm limit was reached for maximum remote connected threads (CONDBAT) (active + inactive).

- QDSTQCIT – the current number of inactive database access threads.
- QDSTQMIT – the maximum number of inactive threads that existed.
- QDSTCNAT – the current number of active database access threads.
- QDSTHWAT – the maximum number of active database access threads that existed.
- QDSTHWDT – the maximum number of active and inactive database access threads that existed.

To monitor the performance of the EDM pool as a result of cacheing, the following fields in IFCID 2 of statistics trace class 1 will be useful:

- QISEDSI – number of inserts into the dynamic statement cache.
- QISEDSG – number of requests for the dynamic statement cache.
- QISEDSC – number of pages used for the dynamic statement cache.

To calculate how often statements are found ready-prepared in the dynamic statement cache, use the formula:

```
hit ratio = (QISEDSG - QISEDSI) / QISEDSG
```

If this ratio is high, and if the overall workload contains heavy use of dynamic SQL statements, many of which execute repeatedly, it indicates that dynamic cacheing is effective.

If the hit ratio is low, it might indicate that dynamic cacheing is not being very effective and the following pros and cons have to be considered.

There is a cost every time DB2 searches the cache and does not find the prepared statement in the cache.

There is a very slight increase in the cost of preparing a

statement that is not already in the cache. There is no benefit if the prepared statement is not executed again before being removed from the cache.

Since both static and dynamic statements share the EDM pool, turning on dynamic statement cacheing adversely affects applications that use static SQL.

It is possible that the few applications that use dynamic statement caching will benefit tremendously when the cache option is justified.

If there are significant numbers of *ad hoc* queries against the database, the hit ratio will be low.

The EDM pool hit ratios for cursor table, package table, and DBD requests must be monitored before and after the dynamic statement cache is enabled. This helps to assess the possible impact of cacheing dynamic SQL statements on the EDM pool usage for static SQL statements, and to make sure that the EDM pool increase is sufficient.

To analyse SQL performance an Explain would be useful. However, with dynamic SQL and reoptimization, a better approach would be to use performance trace Class 3 and IFCID 0022, which has fields similar to some of the columns in the Plan_table. Also, there are other IFCIDs in performance class 3 that would be helpful in capturing information about the actual SQL executed by the Dynamic SQL. These may be used to capture and analyse SQL for performance.

Optimize for '*n*' rows in the context of distributed processing has a different meaning, as follows. Since the CPU cost of sending one row per DRDA network transmission is very high, DB2 assumes that OPTIMIZE FOR 1 ROW is used to force a particular access path, rather than to force a network block size of one row. Hence, when OPTIMIZE FOR 1 ROW is specified, DB2 transmits the lesser of 16 rows and the number of rows that fit within the DRDA query block size on every network transmission. For values of '*n*' greater than 1, DB2 transmits the lesser of '*n*' rows and the number of rows that fit within the DRDA query block size on each network transmission.

CONCLUSION

Distributed database access has become inevitable in enterprises and the above information is presented in order to equip the DBA to respond effectively to those needs. The next step would be to explore tools and mechanisms for capitalizing on this strong capability of DB2 for web-enablement of legacy applications.

REFERENCES

1 *DB2 for OS/390 Version 5 – Administration Guide*.

2 *DB2 for OS/390 Version 5 Performance Topics*, SG24-2213-00 Redbook.

*Jaiwant K Jonathan*
*DB2 DBA*
*QSSI Inc (USA)*



## Interrogating SYSIBM.SYSLGRNX

Some time ago we experienced quite significant DASD hardware errors, which, as you may imagine, upset our DB2 datasharing group. This resulted in the need to recover the majority of our production databases, which was done and all seemed well. Some months later, we hit a problem when we tried to drop and re-create a set of tablespaces and partitions – the DDL failing with Down Level Detection (DLD) messages. We could see that for a given database the *n*th statement failed with this error regardless of the tablespace/partition specified in the DDL. The most obvious reason for this error was that the DBID/PSID combination was already known to DB2, or at least some parts of DB2. To overcome this problem we disabled DLD processing via DSNZPARM until the DDL had run.

Investigating further we came to the conclusion that the part of DB2 which still had a record of the DBID/PSID numbers must be SYSIBM.SYSLGRNX and/or its two indexes. The CHECK utility

 

showed a discrepency and the REBUILD utility corrected it without issue, but prior to carrying out this repair we wanted to load the contents of SYSIBM.SYSLGRNX into a user table so that we could search for other potential errors. Having copied it into our tablespace we hit errors when running SQL against our table because the spacemap page for SYSIBM.SYSLGRNX covers a different range of pages from that of a user tablespace. To overcome this, we decided to write an unload program so that we could use the Load utility to populate our table. The following code contains the actual program plus the JCL to run it, along with some sample SQL statements to interrogate the table. This program was written for DB2 V5.1 and V6.1.

PROGRAM

```
         TITLE 'UNLOAD SYSLGRNX DATA '
*********************************************************************
*   PSEUDOCODE :-    Open files                                     *
*   ----------       read first page                                *
*                    if syslgrnx                                    *
*                    then do                                        *
*                      for each data page                           *
*                      do                                           *
*                        for each id map entry                      *
*                        do                                         *
*                          If id map in use                         *
*                          then do                                  *
*                            write id map value                     *
*                            map syslgrng record                    *
*                            format record and write to output file *
*                          end                                      *
*                        end                                        *
*                      end                                          *
*                    end                                            *
*                    close files                                    *
*                                                                   *
*   REGISTER USE :-  R2   loop control for id map entries          *
*   ------------     R6   base for mapping DB2 page                *
*                    R9   base for mapping syslgrnx record/row     *
*                    R1Ø  internal branching for print routine     *
*                    R12  Base for this program                    *
*                    R13  Pointer to save area                     *
*                                                                   *
*   MODULES :-   None                                              *
*   -------                                                        *
```

```
*                                                                     *
*   MACROS :-                                                         *
*   ------                                                            *
*    ACB      --  Access Control Block                                *
*    CLOSE    --  Close a file                                        *
*    MODID    --  defined internally, used for identification         *
*    DCB      --  Data Communication Block                            *
*    GET      --  Read a record                                       *
*    OPEN     --  Open a file                                         *
*    PUT      --  Write a record                                      *
*    RPL      --  Request Parameter List                              *
*    SHOWCB   --  Get VSAM information                                 *
***********************************************************************
         MACRO
         MODID
.*
.*   descriptive-name = diagnostic identifier generator
.*
         LCLC  &XLABA,&XLABD
&XLABD   SETC  '&SYSPARM'      DIAGNOSTIC LEVEL IF PROVIDED
         AIF   (K'&XLABD GT Ø).SYSPOK
&XLABD   SETC  '&SYSTIME'      DEFAULT=SYSTIME
.SYSPOK  ANOP
         B     DSN&SYSNDX             branch around constant
         DC    CL8'&SYSECT',CL8'&SYSDATE',CL8'&XLABD' DIAGNOSTIC ID
         DC    C'****  '
         DC    C'STEVE KEMP  @ 2ØØ3 '
         DC    C'  ****'
&XLABA   SETC  'DSN&SYSNDX'
&XLABA   DS    ØH
         MEND
         EJECT
*---------------------------------------------------------------------
* entry and set-up
*---------------------------------------------------------------------
DB2LGRNX CSECT
         STM   R14,R12,12(R13)         save caller's registers
         BALR  R12,Ø                   set up base register
         USING *,R12                   ..and tell the assembler
         MODID                         module identifier macro
         ST    R13,SAVEAREA+4          save caller's savearea address
         LA    R11,SAVEAREA            get our savearea address
         ST    R11,8(,R13)             ...and save it
         LR    R13,R11                 save area for called routines
*---------------------------------------------------------------------
* open files and get basic VSAM info
*---------------------------------------------------------------------
OPENØØ   DS    ØH
         OPEN  (READACB)               open VSAM file
         LTR   R15,R15                 did it work
         BNZ   EXITØØ                  ..end if not ok
```

```
         OPEN  (DB2DATA,(OUTPUT))      open db2out dataset
         LTR   R15,R15                 did it work?
         BNZ   CLOSEØ2                 ..end if not ok
         OPEN  (SYSREC,(OUTPUT))       open SYSREC dataset
         LTR   R15,R15                 did it work?
         BNZ   CLOSEØ1                 ..end if not ok
         BAL   R1Ø,PRINIT              initialize print line
         MVC   TEXT2,=C'Checking SYSIBM.SYSLGRNX      '
         BAL   R1Ø,PRNTØ2              print message
         SHOWCB ACB=READACB,                                          +
               AREA=DISPLAY,                                          +
               FIELDS=(NEXT,HALCRBA,ENDRBA),                          +
               LENGTH=12
         MVC   TEXT1,=C'High allocated RBA            '
         L     R1,HALCRBA              get high alloc rba
         BAL   R1Ø,PRNTØØ              ..and print it
         MVC   TEXT1,=C'High used RBA                 '
         L     R1,ENDRBA               get high used rba
         BAL   R1Ø,PRNTØØ              ..and print it
         MVC   TEXT1,=C'Number of extents             '
         L     R1,NEXT                 get number of extents
         BAL   R1Ø,PRNTØØ              ..and print it
*-------------------------------------------------------------------
* read a record (DB2 page)
*-------------------------------------------------------------------
READØØ   DS    ØH
         SR    R9,R9                   clear out r9
         L     R9,=X'ØØØØØØØØ'         set r9 to first page
         ST    R9,RBA                  ..and set rba
         GET   RPL=READRPL             read a record (CI)
         LTR   R15,R15                 did it work?
         BNZ   CLOSEØ1                 ..end if not
         L     R6,RECADDR              point to record
         USING PGHEAD,R6               ..and map it
         LH    R9,HPGDBID              load dbid
         CH    R9,=H'1'                is it the directory?
         BNE   CLOSEØ1                 ..end if not
         LH    R9,HPGPSID              load psid
         CH    R9,=H'2Ø7'              is it syslgrnx?
         BNE   CLOSEØ1                 ..end if not
READØ1   DS    ØH
         MVC   TEXT1,=C' page number                  '
         L     R1,PGNUM                get page number
         BAL   R1Ø,PRNTØØ              ..and print it
         TM    PGFLAGS,B'Ø1111ØØ'      data page?
         BNZ   READØ2                  ..skip if not
         L     R2,PGMAXID              get max id map entries used
         SRL   R2,24                   shift right to truncate
         LTR   R2,R2                   have we any enties
         BH    READØ3                  ..then it is a data page
READØ2   DS    ØH
```

```
         DROP  R6                      free up r6
         L     R9,RBA                  get current rba value
         A     R9,=X'ØØØØ1ØØØ'        increment by db2 page size
         C     R9,ENDRBA               rba past file end?
         BH    CLOSEØ1                 ..then finish
         ST    R9,RBA                  set rba value
         GET   RPL=READRPL             get next record
         LTR   R15,R15                 did it work?
         BNZ   CLOSEØ1                 ..end if not
         L     R6,RECADDR              point to record
         USING PGHEAD,R6               ..and map it
         B     READØ1                  now check it
READØ3   DS    ØH                      this must be a data page
         USING RECORD,R9               set map for record
         SR    R3,R3                   clear out R3
         L     R3,=X'ØØØØØFFC'         point to last entry (fixed)
READØ4   DS    ØH                      loop around id maps
         SR    R7,R7                   clear out r7
         LH    R7,Ø(R3,R6)             load id map entry
         SLL   R7,16                   shift to loose extra stuff
         SRL   R7,16                   ..now shift back
         MVC   TEXT1,=C' id map entry                '
         SR    R1,R1                   clear out r1
         LR    R1,R7                   now set it to idmap value
         BAL   R1Ø,PRNTØ3              ..and print it
         LTR   R7,R7                   is it zero
         BZ    READØ6                  ..skip if yes
         L     R11,=X'ØØØØ8ØØØ'        set flag bit
         CR    R7,R11                  is is free?
         BNL   READØ6                  ..skip if yes
         LR    R9,R6                   point to start of page
         AR    R9,R7                   add offset to record
         L     R8,PGSOBD               ..and load obid
         SRL   R8,16                   shift to the right
         CH    R8,=X'ØØD1'             is it obid x'ØØD1'
         BNE   READØ6                  skip if not
READØ5   DS    ØH                      else process the record
         SR    R11,R11                 clear out r11
         LH    R11,LGRPART             get partition number
         SLL   R11,17                  get rid of high order bit
         SRL   R11,17                  re-align data
         STH   R11,LGRPART             reset partition number
         PUT   SYSREC,DATAREC          write unload record
READØ6   DS    ØH
         S     R3,=F'2'                skip back thru ids
         BCT   R2,READØ4               loop back for next id
         DROP  R9                      free up r9
         B     READØ2                  get the next record
*-------------------------------------------------------------------
* close files
*-------------------------------------------------------------------
```

```
CLOSEØØ  DS    ØH
         CLOSE (SYSREC)               close the ouput file
CLOSEØ1  DS    ØH
         CLOSE (DB2DATA)              close the message file
CLOSEØ2  DS    ØH
         CLOSE (READACB)              close the VSAM file (tablespace)
*-------------------------------------------------------------------
* clear up and exit the program
*-------------------------------------------------------------------
EXITØØ   DS    ØH
         L     R13,4(,R13)            get caller's savearea
         L     R14,12(,R13)           restore register 14
         LM    RØ,R12,2Ø(R13)         restore remaining registers
         BR    R14                    return to caller
*-------------------------------------------------------------------
* print routine
*-------------------------------------------------------------------
PRNTØØ   DS    ØH
         CVD   R1,PACK                convert to decimal number
         MVC   MESSAGE,EDMASK         move edit mask to message
         ED    MESSAGE,PACK+2         edit decimal num with mask
         B     PRNTØ2                 now print it
PRNTØ3   DS    ØH
         ST    R1,ORIG                get id map entry
         UNPK  PRINT(9),ORIG(5)       convert to zoned decimal
         TR    PRINT(8),TRNS-X'FØ'    translate to ascii
         MVC   MESSAGE(4),PRINT+4     copy into rba field
PRNTØ2   DS    ØH
         PUT   DB2DATA,PRNTLINE       print the line
PRINIT   DS    ØH
         MVI   PRNTLINE,C' '          initialize print line
         MVC   PRNTLINE+1(132),PRNTLINE ..to spaces
         MVI   PRNTCNTL,C'-'          set control character
ENDPRNT  DS    ØH
         BR    R1Ø                    go back
*
*-------------------  equates for registers  ----------------------
RØ       EQU    Ø
R1       EQU    1
R2       EQU    2
R3       EQU    3
R4       EQU    4
R5       EQU    5
R6       EQU    6
R7       EQU    7
R8       EQU    8
R9       EQU    9
R1Ø      EQU    1Ø
R11      EQU    11
R12      EQU    12
R13      EQU    13
```

```
R14      EQU    14
R15      EQU    15
*-----------------  program files  -------------------------------
READACB  ACB    DDNAME=SYSUT1,MACRF=(CNV,DIR,SEQ,IN),EXLST=FILEEND,     +
                BUFND=1Ø
READRPL  RPL    ACB=READACB,OPTCD=(CNV,DIR,LOC,NUP),                    +
                AREA=RECADDR,AREALEN=4,ARG=RBA
FILEEND  EXLST  EODAD=CLOSEØ1
DB2DATA  DCB    DSORG=PS,MACRF=PM,DDNAME=DB2OUT,LRECL=133,              +
                RECFM=FBA,BLKSIZE=133
SYSREC   DCB    DSORG=PS,MACRF=PM,DDNAME=SYSREC,LRECL=46,               +
                RECFM=FBA,BLKSIZE=46Ø
*-----------------  constants and variables  ---------------------
SAVEAREA DS     18F                     register save area
DISPLAY  DS     ØF                      VSAM return area
NEXT     DS     F                       VSAM number of extents
HALCRBA  DS     F                       VSAM high allocated rba
ENDRBA   DS     F                       VSAM high used rba
RECADDR  DS     A                       address of vsam buffer
RBA      DS     F                       rba of DB2 page
PACK     DS     D                       conversion for printing
EDMASK   DC     XL12'4Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2121Ø'  edit mask
PRINT    DS     CL9                     ASCII output field
ORIG     DS     CL5                     temporary storage area
TRNS     DC     CL16'Ø123456789ABCDEF'  ASCII translation table
*-----------------  print variables  -----------------------------
PRNTLINE DS     ØCL133
PRNTCNTL DS     C' '
PRNTDATA DS     CL132' '
         ORG    PRNTDATA
MESSAGE  DS     CL12
         DS     CL2
TEXT1    DS     CL3Ø
         DS     CL88
         ORG    PRNTDATA
         DS     CL2Ø
TEXT2    DS     CL3Ø
         DS     CL82
         ORG
*-----------------  dsects  --------------------------------------
PGHEAD   DSECT                          map for header page
PGCOMB   DS     C                       page information
PGLOGRBA DS     CL6                     rba of last update
PGNUM    DS     CL4                     page number
PGFLAGS  DS     C                       flag bits
HPGOBID  DS     ØCL4                    dbid/psid
HPGDBID  DS     CL2                     database id
HPGPSID  DS     CL2                     tablespace id
HPHHPREF DS     CL4                     high preformatted page
HPGCATRL DS     C                       db2 release level
HPGREL   DS     C                       release mark
```

```
FILLØØ   DS    CL2                    reserved
HPGTORBA DS    CL6                    recover to rba
HPGTSTMP DS    CL1Ø                   timestamp fo RO share
HPGSSNM  DS    CL4
         ORG   HPGOBID                remap for data page
PGFREE   DS    CL2                    total free space in page
PGFREEP  DS    CL2                    offset to free space
PGHOLE1  DS    CL2                    offset to large hole
PGMAXID  DS    C                      max number of id map entries
PFPUNCRA DS    C                      possible uncommitted rows
         DS    CL32                   the rest
         ORG
*
RECORD   DSECT
PGSFLAGS DS    C                      flag byte
PGSLTH   DS    CL2                    length of record + 6 byte header
PGSOBD   DS    CL2                    OBID  (x'ØØD1')
PGSBID   DS    C                      id map entry
DATAREC  DS    ØCL46
LGRDBID  DS    CL2                    DBID of entry
LGRPSID  DS    CL2                    PSID of entry
LGRUCDT  DS    CL6                    entry date
LGRUCTM  DS    CL8                    entry time
LGRSRBA  DS    CL6                    start rba
LGRSPBA  DS    CL6                    stop rba
LGRPART  DS    CL2                    ts partition num
LGRSLRSN DS    CL6                    start lrsn
LGRELRSN DS    CL6                    end lrsn
LGRMEMB  DS    CL2                    db2 member id
*
DB2LGRNX CSECT
         END   DB2LGRNX
```

## RUNTIME JCL

```
//STEPØ1 EXEC PGM=DB2LGRNX
//STEPLIB  DD DISP=SHR,DSN=XH17.GEN.LOADLIB
//SYSUT1   DD DISP=SHR,DSN=DB1Ø.DSNDBC.DSNDBØ1.SYSLGRNX.IØØØ1.AØØ1
//DB2OUT   DD SYSOUT=*
//SYSREC   DD DSN=XH17.SYSLGRNX.SYSREC,
//         DCB=(LRECL=46,BLKSIZE=46Ø,RECFM=FB),
//         UNIT=SYSDA,SPACE=(1Ø24,(6ØØ,3ØØ)),DISP=(,CATLG)
//SYSUDUMP DD SYSOUT=*
```

## CREATE JOB

```
//STEPØ1 EXEC PGM=IKJEFTØ1,DYNAMNBR=2Ø
//STEPLIB  DD  DISP=SHR,DSN=SYS1.DB2.LINKLIB
//SYSTSPRT DD  SYSOUT=*
```

```
//SYSTSIN  DD *
  DSN SYSTEM(DB1Ø)
  RUN PROGRAM(DSNTEP2) PLAN(DSNTEP2) -
  LIB('DB1Ø.RUNLIB.LOAD')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD *

-------------------------------------------------
   SET CURRENT SQLID = 'GDB2ADM';
-------------------------------------------------
  DROP TABLESPACE DSNDBØ4.SYSLGRNX;
  COMMIT;
--
  CREATE TABLESPACE SYSLGRNX
  IN DSNDBØ4
  USING VCAT DB1Ø
  FREEPAGE Ø
  PCTFREE Ø
  LOCKSIZE ANY
  BUFFERPOOL BPØ
  CLOSE YES
  ;
--
CREATE TABLE XH17.SYSLGRNX
(LGRDBID          SMALLINT                    ,
 LGRPSID          SMALLINT                    ,
 LGRUCDT          CHAR(6)                     ,
 LGRUCTM          CHAR(8)                     ,
 LGRSRBA          CHAR(6)                     ,
 LGRSPBA          CHAR(6)                     ,
 LGRPART          SMALLINT                    ,
 LGRSLRSN         CHAR(6)                     ,
 LGRELRSN         CHAR(6)                     ,
 LGRMEMB          SMALLINT                    )
IN DSNDBØ4.SYSLGRNX
;
--
CREATE INDEX XH17.SYSLGXØ1
 ON  XH17.SYSLGRNX
 (LGRDBID ASC,
  LGRPSID ASC)
 USING STOGROUP SYSDEFLT
   ERASE NO
   PRIQTY 1ØØØ
   SECQTY  5ØØ
 BUFFERPOOL BPØ
 CLOSE YES
;
//*
```

## LOAD JOB

```
//STARTUT EXEC  PGM=IKJEFTØ1
//SYSPRINT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSIN     DD DUMMY
//SYSTSIN   DD *
  DSN SYSTEM(DB1Ø)
  -TERM UTIL(LOAD.SVK)
  -STA DATABASE(DSNDBØ4) SPACE(SYSLGRNX) ACCESS(UT)
  END
/*
//LOAD   EXEC  PGM=DSNUTILB,PARM='DB1Ø,LOAD.SVK'
//SYSREC   DD  DSN=XH17.SYSLGRNX.SYSREC,DISP=SHR
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR
//SORTWKØ1 DD  UNIT=SYSDA,SPACE=(CYL,(1,1),,CONTIG)
//SORTWKØ2 DD  UNIT=SYSDA,SPACE=(CYL,(1,1),,CONTIG)
//SORTWKØ3 DD  UNIT=SYSDA,SPACE=(CYL,(1,1),,CONTIG)
//SORTWKØ4 DD  UNIT=SYSDA,SPACE=(CYL,(1,1),,CONTIG)
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(1,1),,CONTIG),
//             DISP=(NEW,DELETE,DELETE),
//             DSN=XH17.SYSREC.TEMP2
//SORTOUT  DD  UNIT=SYSDA,SPACE=(CYL,(1,1),,CONTIG),
//             DISP=(NEW,DELETE,DELETE),
//             DSN=XH17.SYSREC1.TEMP2
//DSNTRACE DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//UTPRINT  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSIN    DD  *
LOAD DATA INDDN(SYSREC) LOG(NO) REPLACE
  INTO TABLE XH17.SYSLGRNX
  (LGRDBID     POSITION(1)   SMALLINT,
   LGRPSID     POSITION(3)   SMALLINT,
   LGRUCDT     POSITION(5)  CHAR(6),
   LGRUCTM     POSITION(11) CHAR(8),
   LGRSRBA     POSITION(19) CHAR(6),
   LGRSPBA     POSITION(25) CHAR(6),
   LGRPART     POSITION(31) SMALLINT,
   LGRSLRSN    POSITION(33) CHAR(6),
   LGRELRSN    POSITION(39) CHAR(6),
   LGRMEMB     POSITION(45) SMALLINT
   )
/*
//STARTUT EXEC  PGM=IKJEFTØ1
//SYSPRINT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSIN     DD DUMMY
//SYSTSIN   DD *
  DSN SYSTEM(DB1Ø)
```

```
  -STA DATABASE(DSNDBØ4) SPACE(SYSLGRNX) ACCESS(RW)
  END
/*
```

## RRORG JOB

```
//STOPDB   EXEC  PGM=IKJEFTØ1
//SYSPRINT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSIN     DD DUMMY
//SYSTSIN   DD *
  DSN SYSTEM(DB1Ø)
  -STOP DATABASE(DSNDBØ4) SPACE(SYSLGRNX)
  END
//***************  START TABLESPACE UT **************
//STARTUT EXEC  PGM=IKJEFTØ1
//SYSPRINT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSIN     DD DUMMY
//SYSTSIN   DD *
  DSN SYSTEM(DB1Ø)
  -STA DATABASE(DSNDBØ4) SPACE(SYSLGRNX) ACCESS(UT)
  END
//***************  REORG TABLESPACE *****************
//REORGT EXEC  PGM=DSNUTILB,
//             PARM='DB1Ø,REORG.SYSLGRNX,'
//SORTWKØ1 DD DSN=XH17.SYSLGRNX.SORTWKØ1,DISP=(MOD,DELETE,CATLG),
//     SPACE=(CYL,(15,31),RLSE,,ROUND),UNIT=SYSDA
//SORTWKØ2 DD DSN=XH17.SYSLGRNX.SORTWKØ2,DISP=(MOD,DELETE,CATLG),
//     SPACE=(CYL,(15,31),RLSE,,ROUND),UNIT=SYSDA
//SORTWKØ3 DD DSN=XH17.SYSLGRNX.SORTWKØ3,DISP=(MOD,DELETE,CATLG),
//     SPACE=(CYL,(15,31),RLSE,,ROUND),UNIT=SYSDA
//SORTWKØ4 DD DSN=XH17.SYSLGRNX.SORTWKØ4,DISP=(MOD,DELETE,CATLG),
//     SPACE=(CYL,(15,31),RLSE,,ROUND),UNIT=SYSDA
//SYSREC DD DSN=XH17.SYSLGRNX.REORG1,DISP=(MOD,DELETE,CATLG),
//     SPACE=(CYL,(15,31),RLSE,,ROUND),UNIT=SYSDA
//SYSUT1 DD DSN=XH17.SYSLGRNX.SYSUT1,DISP=(MOD,DELETE,CATLG),
//     SPACE=(CYL,(15,31),RLSE,,ROUND),UNIT=SYSDA
//SORTOUT DD DSN=XH17.SYSLGRNX.SORTOUT,DISP=(MOD,DELETE,CATLG),
//     SPACE=(CYL,(15,31),RLSE,,ROUND),UNIT=SYSDA
//DSNTRACE DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//UTPRINT  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSIN    DD  *
  REORG TABLESPACE DSNDBØ4.SYSLGRNX SORTDATA LOG NO
/*
//*************  IMAGE COPY TABLESPACE **************
//IMAGEC2  EXEC DSNUPROC,UID='XH17.COPYXX',SYSTEM='DB1Ø',
```

```
//             UTPROC=''
//SYSCOPY  DD  DISP=(MOD,CATLG,CATLG),
//             DSN=XH17.DSNDBØ4.SYSLGRNX.FIC(+1),
//             DCB=(BLKSIZE=4Ø96,BUFNO=1ØØ),
//             UNIT=339Ø,LABEL=(,,,,RETPD=31),
//             SPACE=(CYL,(5,5),RLSE)
//DSNTRACE DD  SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//SYSUDUMP  DD DUMMY
//SYSIN     DD *
  COPY TABLESPACE DSNDBØ4.SYSLGRNX COPYDDN SYSCOPY SHRLEVEL REFERENCE
/*
//***************  RUNSTATS TABLESPACE  ***********
//RUNSTAT EXEC  PGM=DSNUTILB,PARM='DB1Ø,STATS.RUNST,'
//SYSREC   DD  DUMMY
//DSNTRACE DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSIN    DD *
  RUNSTATS TABLESPACE DSNDBØ4.SYSLGRNX
  INDEX ALL TABLE ALL REPORT YES UPDATE ALL SHRLEVEL REFERENCE
/*
//***************  START TABLESPACE UT **************
//STARTRW EXEC  PGM=IKJEFTØ1
//SYSPRINT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSIN     DD DUMMY
//SYSTSIN   DD *
  DSN SYSTEM(DB1Ø)
  -STA DATABASE(DSNDBØ4) SPACE(SYSLGRNX) ACCESS(RW)
  END
/*
```

## QUERY JOB

```
//STEPØ1 EXEC PGM=IKJEFTØ1,DYNAMNBR=2Ø
//STEPLIB  DD  DISP=SHR,DSN=SYS1.DB2.LINKLIB
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD *
  DSN SYSTEM(DB1Ø)
  RUN PROGRAM(DSNTEP2) PLAN(DSNTEP2) -
  LIB('DB1Ø.RUNLIB.LOAD')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD *
   SET CURRENT SQLID = 'GDB2ADM';
    SELECT A.DBNAME, A.NAME, A.DBID, A.PSID
    FROM SYSIBM.SYSTABLESPACE A
    WHERE NOT EXISTS (
```

```
            SELECT B.LGRDBID,
                   B.LGRPSID
            FROM XH17.SYSLGRNX B
            WHERE B.LGRDBID = A.DBID
            AND   B.LGRPSID = A.PSID
            )
    ORDER BY 3 ASC, 4 ASC
    ;
    SELECT * FROM XH17.SYSLGRNX
    WHERE LGRDBID = 281
    AND LGRPSID = 2
    ;
    SELECT A.LGRDBID, A.LGRPSID
    FROM XH17.SYSLGRNX A
    WHERE NOT EXISTS (
            SELECT B.DBID,
                   B.PSID
            FROM SYSIBM.SYSTABLESPACE B
            WHERE A.LGRDBID = B.DBID
            AND   A.LGRPSID = B.PSID
            )
    ORDER BY 1 ASC, 2 ASC
    ;
/*
```

*Steve Kemp*
*(UK)*


## ISPF dialog to obtain a list of DB2 subsystems

WHICH DB2 SUBSYSTEMS ARE DEFINED ON THIS MVS LPAR?

Many ISPF dialogs need to know which DB2 subsystems are defined in the MVS logical partition (LPAR) where the dialog is executing. This information can be used by the ISPF dialogs to determine the parameter to be used to establish an attachment to a specific DB2 subsystem.

Traditionally, ISPF dialogs rely on hard-coded logic to validate which DB2 subsystem name is acceptable as a parameter to be used when connecting to DB2. Now that MVS Sysplexes and

 

DB2 data-sharing groups are more widely used, the combinations of multiple MVS LPARs and DB2 subsystems have increased the complexity of validating which DB2 subsystems names are acceptable to use in any specific MVS LPAR. For example, it is typical of data sharing set-ups to define all DB2 members in all MVS LPARs, so a member can be restarted in any of the other MVS LPARs of the Sysplex.

A list of active DB2 subsystems is of even more interest than the list of defined DB2 subsystems. Obtaining such a list will ease development of ISPF applications and facilitate a better user interface by informing the user which DB2 subsystems are available to attach to.

## OBTAINING A LIST OF ACTIVE DB2 SUBSYSTEMS

One possible approach to finding a list of active DB2 subsystems in an MVS LPAR is to find the active IEFSSN member in SYS1.PARMLIB where all subsystems are defined to the OS/390 operating system, parse its contents, and build a list of the defined DB2 subsystems. If using a parsing approach, the process will also need to verify whether the subsystems are active, usually with a test connection against each one of the defined DB2 subsystems; the results of this test will indicate whether the DB2 subsystem is active or not, and a list of active systems can be composed from these results. This approach is feasible but it is certainly not elegant.

A more complex approach to obtaining the list of active DB2 subsystems is to examine the MVS, JES2, and DB2 control blocks in memory at the time the ISPF dialog executes, determine which DB2 subsystems are active from these control blocks, and build a list with their names.

## ISPF DIALOG FOR A LIST OF ACTIVE AND INACTIVE SUBSYSTEMS

The purpose of this article is to provide an ISPF dialog that will build two lists – one of active and the other of inactive DB2 subsystems in the MVS LPAR where this REXX program is

executed. This ISPF dialog comprises two REXX programs (DRIVER and DB2LIST) and an ISPF panel.

## DB2LIST PROGRAM LOGIC

The logic for the DB2LIST program uses the fact that DB2 systems are defined as MVS subsystems. MVS builds memory control blocks for the different subsystems (such as DB2, CICS, MQ, etc) that are defined in the active IEFSSN*xx* member in SYS1.PARMLIB. Certain subsystems such as DB2 require a special control block named ERLY (Early Processing block), which gets built by MVS, even if the DB2 subsystem is not activated; in other words, all defined DB2 subsystems will have an ERLY control block in memory.

The ERLY control block contents can be found in member DSNDCBDS of the DB2 SDSNSAMP partitioned dataset (PDS). This control block contains general subsystem information, such as the name of the DB2 subsystem, the DB2 subsystem recognition character, the ERLY code module name, the DB2 component ID base number, and many other fields including an address to the SCOM control block. ERLY control blocks contain a value of DSN3EPX in the ERLYMODN field; this value can be used to identify which ERLY control blocks belong to a DB2 subsystem.

The SCOM (subsystem communication) control block gets built **only** after a DB2 subsystem is started and it goes away after a DB2 subsystem is stopped. So while a DB2 subsystem is active, the SCOM block exists in memory, and thus the ERLYSCOM address field in the ERLY control block contains a non-zero memory address.

The relationship between the MVS, JES2, and DB2 control blocks is described in chapter 36 of the DB2 Version 7 *Diagnosis Guide and Reference* (IBM publication LY37-3740-00). Page 719 contains a diagram depicting which MVS, JES2, and DB2 control blocks are connected. Combining this diagram with the contents of the DSNDCBDS member and the MVS data areas

provides all the information required for creating a list of active and inactive DB2 subsystems.

With that in mind, the DB2LIST REXX program performs the following:

1 Gets the address of the Communications Vector Table (CVT), which can be found at offset X'10' of the Prefixed Save Area (PSA) MVS memory block. The PSA maps the storage that starts at location X'00' in memory for the related processor, and it contains addresses to many MVS control blocks.

2 Gets the address of the JES Job Control Table (JCT), which can be found at offset X'128' off the CVT block. The JCT is the primary job-oriented control block in the Job Entry Subsystem (JES).

3 Gets the address of the first Subsystem Communications Vector table (SSCVT) block that can be found at offset X'18' of the JCT block. If the SSCVT address is not zeroes (it is zeroes when the end of the list of memory blocks is reached), then we need to know if there is an ERLY block associated with the SSCVT block.

4 Gets the address of the ERLY block at offset X'14' of the SSCVT block. Since there are subsystems that do not have an ERLY block associated with them, we need to verify that the address is really pointing to an ERLY block; we do so by looking for the ‘ERLY’ constant eye-catcher at offset X'4' of the ERLY address.

5 If it is an ERLY block, we need to determine whether it belongs to a DB2 subsystem. We do so by verifying that the field ERLYMODN matches the DSN3EPX program name used by DB2 for ERLY definition.

6 If it is a DB2 ERLY block, then we also need to determine whether the ERLYSCOM field contains zeroes or not. Zeroes are an indication that the DB2 subsystem is inactive; non-zero values will be the address of the SCOM block of an active DB2 subsystem.

7 If it is an active DB2 subsystem, add the name of the DB2 subsystem to a list of active DB2 subsystems. Likewise, if it is an inactive DB2 subsystem, add its name to the list of inactive DB2 subsystems.

8 Get the address of the next SSCVT control block and repeat items 4 to 7. Whenever the SSCVT address is zeroes, it will mean that we have traversed the list of all known subsystems in the present MVS LPAR.

9 We have built a list of active DB2 subsystems; let's see it!

## DRIVER REXX PROGRAM

The purpose of this REXX program is to invoke the DB2LIST REXX program and display the results on an ISPF panel. It also illustrates how to separate the results generated by the DB2LIST program into two separate lists – the DB2LIST (active DB2 subsystems lists) and the DB2INACT (inactive DB2 subsystem lists).

```
/* REXX */
Numeric Digits 256         /* this must be at top of REXX.*/
/*-------------------------------------------------------*/
/* call the DB2LIST rexx program with the NOPRINT parm.  */
/*-------------------------------------------------------*/
ALLLIST = DB2LIST(NOPRINT) /* call the DB2LIST REXX pgm   */
/*-------------------------------------------------------*/
/*- split the ALLLIST variable into two lists using the  */
/*- '=' sign as the delimiter between the two lists.     */
/*-------------------------------------------------------*/
PARSE VAR ALLLIST ACTLIST "=" INACTLIST /* split result  */
/*-------------------------------------------------------*/
/* allocate and display panel.  Then deallocate user lib */
/*-------------------------------------------------------*/
ADDRESS ISPEXEC "LIBDEF ISPPLIB DATASET ID('USER.ISPPLIB')"
ADDRESS ISPEXEC "DISPLAY PANEL(PDB2LIST)"
ADDRESS ISPEXEC "LIBDEF ISPPLIB"
EXIT
```

## DB2LIST REXX PROGRAM

As mentioned before, this REXX program looks into the different MVS, JES2, and DB2 memory blocks and creates a list of active

and inactive DB2 subsystems. Depending on the parameter used, it will either print the lists or return them as a result to the caller.

```
/* rexx */
/*********************************************************************/
/*      Build a list of active and inactive DB2 subsystems.          */
/*      Return lists or display lists depending on parameter passed */
/*********************************************************************/
Numeric Digits 256            /* this must be at top of REXX.      */
PARSE ARG SAYFLAG
ALLLIST = BUILD_DB2SSN_LIST()
/*-------------------------------------------------------------------*/
/* check SAYFLAG parameter to see if results are printed or not.    */
/*-------------------------------------------------------------------*/
if SAYFLAG = 'NOPRINT' then return ALLLIST
/*-----------------------------------------------------------*/
/*  split the ALLLIST variable into two lists using the '='   */
/*  sign as the delimiter between the two lists.              */
/*-----------------------------------------------------------*/
PARSE VAR ALLLIST ACTLIST "=" INACLIST
/*-----------------------------------------------------------*/
/*  display id. Message and list of active and inactive db2s. */
/*-----------------------------------------------------------*/
Say "DB2LIST REXX V1.Ø executing at  :" SYSID
Say "List of Active   DB2 systems is :" ACTLIST
Say "List of Inactive DB2 systems is :" INACLIST
Exit
/*********************************************************************/
/*      Sub-routine build_db2ssn_list:  part of DB2LIST REXX pgm.   */
/*      Build a list of active DB2 subsystems.  Return a single    */
/*      variable string, containing both list of active and inactive*/
/*      db2 subsystems.  DB2s names are comma separated. The two   */
/*      lists (active and inactive) are separated by the '=' sign.  */
/*********************************************************************/
build_db2ssn_list:
Numeric Digits 256            /* this must be set all times        */
CVTADDR = Get_Stor('1Ø'x)     /* get communications vector address */
SYSID   = Strip(Get_Stor(CVTADDR,'154'x,8))  /* get MVS system id.  */
/*-------------------------------------------------------------------*/
/*      Traverse the list of subsystems known to JES.               */
/*      JESCT = CVT+x'128' = Pointer to the JESCT                    */
/*      JESCT points to first SSCT which chains to next SSCT        */
/*-------------------------------------------------------------------*/
JCTADDR  = Get_Stor(CVTADDR,'128'x) /* JES job control table block  */
SSCTADDR = Get_Stor(JCTADDR,24)     /* Address of first SSCT block  */
ACTLIST = ''                        /* initialize variable to nil.  */
INACLIST = ''                       /* initialize variable to nil.  */
/*-------------------------------------------------------------------*/
/*      Top of the loop for traversing the chain of SSCT blocks.    */
```

```
/*-------------------------------------------------------------*/
SSCTLOOP:
/*-------------------------------------------------------------*/
/*      get information from the SSCT control block.           */
/*-------------------------------------------------------------*/
SUBSYS  = Get_Stor(SSCTADDR,'8'x)   /* get the Subsystem name.     */
ERLYADDR= Get_Stor(SSCTADDR,2Ø)     /* get ERLY block address.     */
/*-------------------------------------------------------------*/
/*      get information from the ERLY control block.           */
/*-------------------------------------------------------------*/
ERLYEYEC= Get_Stor(ERLYADDR,4)      /* get literal ERLY eye catcher */
ERLYMODN= Get_Stor(ERLYADDR,84,8)   /* get name of module associated*/
SCOMADDR= Get_Stor(ERLYADDR,56)     /* get SCOM block address.     */
/*-------------------------------------------------------------*/
/*      verify that it is an ERLY block and it is for DB2.     */
/*-------------------------------------------------------------*/
if ERLYEYEC = 'ERLY' & ERLYMODN = 'DSN3EPX ' THEN DO
   /*----------------------------------------------------------*/
   /*   If the SCOM address is zeroes, then DB2 is not active.*/
   /*----------------------------------------------------------*/
   if C2D(SCOMADDR) = Ø then do
      if INACLIST = '' then INACLIST = SUBSYS
      else INACLIST = INACLIST || ',' || SUBSYS
      end
   else do
      /*-------------------------------------------------------*/
      /*  if system is active, then add it to the subsystem list.*/
      /*-------------------------------------------------------*/
      if ACTLIST = '' then ACTLIST = SUBSYS
      else ACTLIST = ACTLIST || ',' || SUBSYS
      end
end
/*-------------------------------------------------------------*/
/* get address of next SSCT control block in the chain.        */
/* if the address is not zeroes, then there is another SSCT, branch */
/* back to the top of the loop at label SSCTLOOP.              */
/*-------------------------------------------------------------*/
SSCTADDR = Get_Stor(SSCTADDR,'4'x)            /* Next one in chain */
If C2D(SSCTADDR) ¬= Ø Then Signal SSCTLOOP
/*-------------------------------------------------------------*/
/* last SSCTADDR was found to be zeroes, so that was the last one.  */
/* combine the two db2 lists so a single value can be returned.     */
/*-------------------------------------------------------------*/
Return ACTLIST || '=' || INACLIST
Get_Stor: PROCEDURE
/*-------------------------------------------------------------*/
/*      This procedure will extract data using the MVS Storage */
/*      REXX function. Input arguments will be:                */
/*      1) Storage_Pointer or Literal, ie CVTPTR or '1Ø'x, or  */
/*         16 (like '1Ø'x)                                     */
```

```
/*      2) Offset in hex or dec (number), ie 'FF'x or 256 or       */
/*         D2C(256)                                                */
/*      3) Length of returned data in decimal, ie 256              */
/*      NOTE – ensure that a 'NUMERIC DIGITS 256' is at            */
/*             the beginning of the calling REXX program.          */
/*-----------------------------------------------------------------*/
Parse Arg AREA,OFFSET,LENG
If Arg(2,'O') Then OFFSET=Ø
If Arg(3,'O') Then LENG=4
If DataType(AREA)   = 'CHAR' Then AREA   = C2D(AREA)
If DataType(OFFSET) = 'CHAR' Then OFFSET = C2D(OFFSET)
Return  Storage((D2X(AREA+OFFSET)),LENG)
```

## ISPF PANEL PDB2LIST

This ISPF panel can be used to display the results of the DB2LIST REXX program. This panel needs to be copied to any PDS and the DRIVER REXX program will need to be updated to reflect the panel location.

```
)ATTR
/*************************************************************/
/* PANEL: PDB2LIST – DISPLAY RESULTS OF DB2LIST REXX PROGRAM  */
/*************************************************************/
 % TYPE(TEXT)   INTENS(HIGH)   COLOR(WHITE)
 @ TYPE(TEXT)   INTENS(HIGH)   COLOR(YELLOW)
 + TYPE(TEXT)   INTENS(LOW)    COLOR(TURQ)  SKIP(ON)
 ~ TYPE(TEXT)   INTENS(HIGH)   COLOR(BLUE)
 ! TYPE(OUTPUT) INTENS(LOW) HILITE(REVERSE)  COLOR(GREEN)
 # TYPE(INPUT)  INTENS(LOW)                  COLOR(RED)
 _ TYPE(INPUT)  INTENS(LOW) HILITE(USCORE)   COLOR(RED)
)BODY
%(PDB2LIST) -------- DISPLAY LIST OF ACTIVE DB2 SYSTEMS --------
%COMMAND ===>#ZCMD
@
%   Please enter the following information :
%
% +Select DB2 system from:!ACTLIST                              +
%
%  +Enter DB2 system  ===>_Z   +
%
%  +List of Inactive DB2s:!INACLIST                             +
%
%  +Press~PF3+to exit this panel
)INIT
 .ZVARS = '( DB2SSN )'
 .CURSOR = ZCMD
 IF (&ACTLIST = &Z)
    &ACTLIST = 'ERROR_IN_DB2LIST_REXX_PROGRAM'
```

```
 IF (&DB2SSN = &Z )
    &DB2SSN = TRUNC(&ACTLIST,'.')
)REINIT
 .MSG = &Z
 REFRESH(*)
 IF (&DB2SSN = &Z )
    &DB2SSN = TRUNC (&ACTLIST,'.')
)PROC
 REFRESH(*)
 IF ( .RESP = ENTER )
    VER (&DB2SSN,NAME)
    VER (&DB2SSN,NB)
    &ZEDSMSG = 'INVALID DB2 SYSTEM'
    &ZEDLMSG = 'CHOOSE FROM: &ACTLIST'
    VER (&DB2SSN,LISTV,&ACTLIST,MSG=ISRZ001)
 IF (.MSG = &Z)
    VPUT  ( DB2SSN )
)END
```

Assuming there are active DB2 subsystems, the &ACTLIST variable should have been populated by the DRIVER REXX program. In this panel, the &ACTLIST variable contains either the literal 'ERROR_IN_DB2LIST_REXX_PROGRAM' or a list of active DB2 systems.

The heart of this panel can be found in the VER (&DB2SSN,LISTV,&ACTLIST,..) statement. This statement allows the use of a variable (&ACTLIST) containing a list of DB2 subsystem names to be used for variable field verification. The input variable &DB2SSN will be checked against the &ACTLIST variable and, if a match is not found, the VER statement will set the MSG control variable to ISRZ001, causing the error messages to be displayed. Additional information on the VER statement and its LISTV clause can be found on pages 252-263 of the *OS/390 V2R10.0 ISPF Dialog Developer's Guide and Reference*.

## CALLING THE DB2LIST REXX PROGRAM DIRECTLY

If the DB2LIST REXX program is stored in one of the partitioned datasets allocated to your TSO JCL logon procedure ISPCLIB or ISPEXEC DDNAME statements, you should be able to call it directly by issuing the TSO %DB2LIST command.

CLOSING

Having the capability to find which DB2 subsystems are active on the current MVS LPAR can be easily accomplished with the DB2LIST REXX program. This capability should help simplify the creation of DB2-related ISPF dialogs.

*Antonio Luis Salcedo Freidel*
*Lead DB2 Systems Programmer (USA)*


## Identity column

DB2 V7 has introduced a new concept that can guarantee unique column values without having to create an index. You can eliminate the application coding that was implemented to assign unique column values for those columns. The AS IDENTITY option in CREATE TABLE or ALTER TABLE specifies that the column is an identity column for the table. An identity column is a numeric, either SMALLINT, INTEGER, or DECIMAL, with a scale of zero or a user-defined distinct type based on any of these data types, which is UNIQUE and NOT NULL by definition. The support for identity columns provides a way to have DB2 automatically generate unique, sequential, and recoverable values for the column defined as the identity column for each row in the table. Duplicate values are possible for an identity column if you specify the CYCLE option. A table can have no more than one identity column.

In our company, we move (replicate) many tables from the mainframe to the local servers. Our problem is that in many cases we replicate all data (Import or Load utility). This movement takes a lot of time, so we decided to **alter tables** with additional identity columns or **alter tables** with additional timestamp columns defined like NOT NULL WITH DEFAULT.

THE FIRST SOLUTION

Each table that is a candidate for propagation is altered with an

identity column. This column is unique and will be used later in a trigger definition.

Here is an example on installation sample table DSN8710.EMP:

```
ALTER TABLE DSN871Ø.EMP ADD ACOLUMN_ID INTEGER
            GENERATED ALWAYS AS IDENTITY
            (START WITH 1,
             INCREMENT BY 1,
             CACHE 2Ø, NO CYCLE);
```

When you add an identity column to a table that is not empty, DB2 places the table space that contains the table in the REORG pending state. When the REORG utility is run, DB2 generates the values for the identity column in all existing rows and then removes the REORG pending status. The REORG utility requires the COPYDDN option.

The next step defines triggers on the DSN8710.EMP table. You must create an INSERT trigger, UPDATE trigger, and DELETE trigger. Then all triggers insert data to the specific log table, which must be already defined. The CREATE statement for log table INFO.LOGI is:

```
CREATE TABLESPACE SSTRGØ1 IN DBTRIGER
       USING      STOGROUP STGØ1
                    PRIQTY 8ØØ
                    SECQTY 8ØØ
                     ERASE NO
                  FREEPAGE 4
                   PCTFREE 5
       BUFFERPOOL BP3
         LOCKSIZE ANY
            CLOSE YES
         COMPRESS NO
            CCSID EBCDIC
          LOCKMAX SYSTEM
          SEGSIZE 64;
CREATE TABLE INFO.LOGI
   (ZAP                  INTEGER
                         GENERATED ALWAYS AS IDENTITY
                         (START WITH 1,
                          INCREMENT BY 1,
                          CACHE 2Ø, NO CYCLE,
                          MAXVALUE 2147483647,
                          MINVALUE 1)
   ,TIM                  TIMESTAMP             NOT NULL WITH DEFAULT
```

```
   ,TAB                VARCHAR(18)          NOT NULL
   ,CRE                CHAR(8)              NOT NULL
   ,KEY                INTEGER              NOT NULL
   ,AKC                CHAR(1)              NOT NULL
   ,USR                CHAR(8)              NOT NULL DEFAULT USER
   ) IN DBTRIGER.SSTRGØ1
   CCSID EBCDIC;
ALTER TABLE INFO.LOGI
      ADD   CONSTRAINT CAKC
            CHECK (AKC IN ('I', 'U', 'D', 'T')
                  );
CREATE  INDEX INFO.XCRE
              ON INFO.LOGI
       ( CRE ASC
       , TAB ASC
       , TIM ASC )
       USING      STOGROUP STGØ2
                    PRIQTY 12Ø
                    SECQTY 12Ø
                     ERASE NO
                  FREEPAGE 2
                   PCTFREE 5
       BUFFERPOOL BP1
            CLOSE YES
             COPY NO
        PIECESIZE 2Ø97152 K;
```

Description of the table attributes:

- ZAP – identity column
- TIM – timestamp
- TAB – table name
- CRE – creator table name
- KEY – relations on attribute ACOLUMN_ID
- AKC – action (I-Insert, U-Update, D-Delete, T-Transfer)
- USR – user id.

The INFO.LOGI table contains change information (rows) for all tables that are defined with triggers. In this table are keys for all modification rows for a specific table and in the next step you can propagate only changed rows to the local servers.

Example:

```
SELECT E.ACOLUMN_ID
FROM DSN871Ø.EMP E,
     INFO.LOGI L
WHERE E.ACOLUMN_ID = L.KEY
AND DATE(L.TIM) = CURRENT DATE – 1 DAY
WITH UR
```

The query returns all keys from table DSN8710.EMP that have been modified ‘current date - 1day’.

## ICOL – REXX DRIVER PROCEDURE

```
/* REXX */
/* TRACE R */
 zpfctl = 'OFF'
 Y=MSG("OFF")
 /* DSNREXX Language Support                                  */
 Address TSO "SUBCOM DSNREXX"
 IF RC THEN
 S_RC = RXSUBCOM(ADD,DSNREXX,DSNREXX)
 Top:
 address ispexec "display panel(ICOLM)"
 if rc=8 then Exit
 vol1=Ø; vol2=Ø; vol3=Ø
 if serv1<>'' then vol1=1
 if serv2<>'' then vol2=1
 if serv3<>'' then vol3=1
 SSID = db2
 ADDRESS DSNREXX "CONNECT" SSID
 SQLSTMT="SELECT STRIP(P.VCATNAME),STRIP(P.DBNAME),STRIP(P.TSNAME), ",
 "        CASE MAX(PARTITION)                                       ",
 "         WHEN Ø THEN 1                                            ",
 "         ELSE MAX(PARTITION)                                      ",
 "        END, SUBSTR(CHAR(CURRENT TIMESTAMP),21,6)                 ",
 " FROM SYSIBM.SYSTABLES T,                                         ",
 "      SYSIBM.SYSTABLEPART P                                       ",
 " WHERE CREATOR='"cre"'                                            ",
 "   AND NAME='"tab"'                                               ",
 "   AND T.TYPE='T'                                                 ",
 "   AND T.DBNAME=P.DBNAME                                          ",
 "   AND T.TSNAME=P.TSNAME                                          ",
 " GROUP BY P.VCATNAME, P.DBNAME, P.TSNAME                          ",
 " WITH UR                                                          "
 Address DSNREXX "EXECSQL DECLARE C1 CURSOR FOR S1"
 Address DSNREXX 'EXECSQL PREPARE S1 FROM :SQLSTMT'
```

```
Address DSNREXX "EXECSQL OPEN C1"
Address DSNREXX,
 "EXECSQL FETCH C1 INTO :HVCAT, :HVDB, :HVTS, :HVPA, :HVTG"
if sqlcode=100 then do
   zedsmsg = "Table not found"
   zedlmsg = "Table not found. Enter table name."
   Address DSNREXX "EXECSQL CLOSE C1"
   address ispexec "setmsg msg(isrz001)"
   signal top
end
if sqlcode<>0 then do
   zedsmsg = 'Sqlcode '||sqlcode
   zedlmsg = 'Error. Sqlcode '||sqlcode
   Address DSNREXX "EXECSQL CLOSE C1"
   address ispexec "setmsg msg(isrz001)"
   signal top
end
pri=0; sec=0
do i=1 to hvpa
   dsn=hvcat||'.DSNDBD.'||HVDB||'.'||HVTS||'.I0001.A'
   part=right(i,3,'0')
   dsn=dsn||part
   dsn = "('"dsn"')"
   X=OUTTRAP('var.')
   address tso "listc" entries dsn allocation
   X=OUTTRAP('OFF')
   hurba = word(translate(var.9,' ','-'),7)
   if hurba < trunc(737280/15,0) then do
      prip=1
      secp=1
   end
   else do
      prip=trunc((hurba/(737280/15)+1),0)
      secp=max(trunc(prip*0.05,0),1)
   end
   pri=pri+prip
   sec=sec+secp
end
tid='COLUMN'
Address DSNREXX "EXECSQL CLOSE C1"
/* JCL Skeleton Alter Table Log Key                    */
dsufB='D'||right(date('D'),3,'0')||right(time('M'),4,'0')
date=date()
time=time(c)
user=userid()
tempfile=userid()||'.UTIL.LOG.KEY'
address tso
"delete '"tempfile"'"
"free dsname('"tempfile"')"
```

```
 "free ddname(ispfile)"
 "free atttlist(formfile)"
 "attrib formfile blksize(800) lrecl(80) recfm(f b) dsorg(ps)"
 "alloc ddname(ispfile) dsname('"tempfile"')",
        "new using (formfile) unit(3390) space(1 1) cylinders"
 address ispexec
 "ftopen"
 "ftincl ICOLS"
 "ftclose"
 zedsmsg = "JCL shown"
 zedlmsg = "JCL ATLC Utility shown"
 "setmsg msg(isrz001)"
 "edit dataset('"tempfile"')"
 exit
```

## ICOLM – ENTRY PANEL

```
)ATTR
   $ type(text)   color(white) caps (off) hilite(reverse) intens(high)
   | type(text)   color(white)     hilite(reverse) intens(high)
   ( type(text)   color(yellow)    hilite(reverse) intens(high)
   ) type(text)   color(green)                     intens(high)
   _ type(input)  color(red)       intens(high) pad(_)
)BODY WINDOW(36,18)
+
$      Alter Table Log Key
| +                                |
| )  Db2     :_db2 +               |
| )  Creator :_cre     +           |
| )  Table   :_tab                +|
| )  Index   :_inx+                |
| )  Runstat :_rst+                |
| )  Trigger :_trg+                |
| )  Volume  :_serv1 +             |
| )          :_serv2 +             |
| )          :_serv3 +             |
| +                                |
( Enter:Continue            PF3:End
)INIT
  if (&db2 ¬= ' ')
      .attr (db2) = 'pad(nulls)'
  if (&cre ¬= ' ')
      .attr (cre) = 'pad(nulls)'
  if (&tab ¬= ' ')
      .attr (tab) = 'pad(nulls)'
  if (&inx ¬= ' ')
      .attr (inx) = 'pad(nulls)'
  if (&rst ¬= ' ')
      .attr (rst) = 'pad(nulls)'
```

```
  if (&trg ¬= ' ')
      .attr (trg) = 'pad(nulls)'
  if (&serv1¬= ' ')
      .attr (serv1) = 'pad(nulls)'
  if (&serv2¬= ' ')
      .attr (serv2) = 'pad(nulls)'
  if (&serv3¬= ' ')
      .attr (serv3) = 'pad(nulls)'
)PROC
  &inx = TRANS(TRUNC(&inx,1) Y,YES N,NO)
  VER(&inx LIST YES,NO)
  VER(&inx,NONBLANK)
  &rst = TRANS(TRUNC(&rst,1) Y,YES N,NO)
  VER(&rst LIST YES,NO)
  VER(&rst,NONBLANK)
  &trg = TRANS(TRUNC(&trg,1) Y,YES N,NO)
  VER(&trg LIST YES,NO)
  VER(&trg,NONBLANK)
  IF (.PFKEY = PFØ3) &PF3 = EXIT
  VPUT (tab cre db2 inx rst trg serv1 serv2 serv3) PROFILE
)END
```

## ICOLC – SKELETON JCL

```
)TBA 72
)CM -------------------------------------------------------------------
)CM Skeleton: Alter Table Log Key Utility                            --
)CM -------------------------------------------------------------------
//&user.X JOB (777-ICOL),'ICOL',
//             NOTIFY=&user,REGION=4M,
//             CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//* ******************************************************************
//RUNSQL1 EXEC PGM=IKJEFTØ1
//STEPLIB DD DISP=SHR,DSN=DSN71Ø.SDSNLOAD
//        DD DISP=SHR,DSN=CEE.SCEERUN
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(&db2)
 RUN PROGRAM(DSNTIAD) PLAN(DSNTIA71) -
     LIB('DSN71Ø.RUNLIB.LOAD.DSNN')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD *
  ALTER TABLE &cre..&tab ADD A&tid._ID INTEGER
                    GENERATED ALWAYS AS IDENTITY
                    (START WITH 1,
                     INCREMENT BY 1,
                     CACHE 2Ø, NO CYCLE);
  COMMENT ON COLUMN &cre..&tab..A&tid._ID IS 'IDENTITY KOLONA';
```

```
  COMMIT;
)SEL &inx EQ YES
  CREATE UNIQUE INDEX INFO.X&hvtg
         ON &cre..&tab
          ( A&tid._ID DESC )
         USING STOGROUP GLLMØ6
                PRIQTY 1ØØ
                SECQTY 1ØØ
              FREEPAGE 5
               PCTFREE 1Ø
         BUFFERPOOL BP3 ;
)ENDSEL
//*---- TERMINATE UTILITY ---------------------------
//TERMU   EXEC PGM=IKJEFTØ1,COND=(4,LT)
//STEPLIB  DD DSN=DSN71Ø.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(DSNN)
   -TERM UTILITY(&user..REORC)
   END
/*
//*---- REORG &hvdb..&hvts
//REOR1  EXEC DSNUPROC,SYSTEM=&db2,REGION=4Ø96K,
//      UID='&user..REORC',UTPROC=''
//STEPLIB DD DSN=DSN71Ø.SDSNLOAD,DISP=SHR
)SEL &vol1 EQ Ø
//SYSREC DD UNIT=SYSDA,
)ENDSEL
)SEL &vol1 EQ 1
//SYSREC DD UNIT=339Ø,VOL=SER=&serv1,
)ENDSEL
//    DSN=&user..&hvdb..&hvts..REC.&dsufb,
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,CATLG)
)SEL &vol1 EQ Ø
//SORTOUT DD UNIT=SYSDA,
)ENDSEL
)SEL &vol1 EQ 1
//SORTOUT DD UNIT=339Ø,VOL=SER=&serv1,
)ENDSEL
//    DSN=&user..&hvdb..&hvts..OUT.&dsufb,
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,CATLG)
)SEL &vol1 EQ Ø
//SYSUT1 DD UNIT=SYSDA,
)ENDSEL
)SEL &vol1 EQ 1
//SYSUT1 DD UNIT=339Ø,VOL=SER=&serv1,
)ENDSEL
//    DSN=&user..&hvdb..&hvts..UT1.&dsufb,
```

```
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,CATLG)
)SEL &vol1 EQ Ø
//SYSCOPY DD UNIT=SYSDA,
)ENDSEL
)SEL &vol1 EQ 1
//SYSCOPY DD UNIT=339Ø,VOL=SER=&serv1,
)ENDSEL
//    DSN=&user..&hvdb..&hvts..SYSCOPY.&dsufb,
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(MOD,CATLG,CATLG)
)SEL &vol2 EQ Ø
//SORTWKØ1 DD UNIT=SYSDA,
)ENDSEL
)SEL &vol2 EQ 1
//SORTWKØ1 DD UNIT=339Ø,VOL=SER=&serv2,
)ENDSEL
//    DSN=&user..&hvdb..&hvts..WK1.&dsufb,
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,DELETE),
//    DCB=(BUFNO=1Ø)
)SEL &vol2 EQ Ø
//SORTWKØ2 DD UNIT=SYSDA,
)ENDSEL
)SEL &vol2 EQ 1
//SORTWKØ2 DD UNIT=339Ø,VOL=SER=&serv2,
)ENDSEL
//    DSN=&user..&hvdb..&hvts..WK2.&dsufb,
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,DELETE),
//    DCB=(BUFNO=1Ø)
)SEL &vol3 EQ Ø
//SORTWKØ3 DD UNIT=SYSDA,
)ENDSEL
)SEL &vol3 EQ 1
//SORTWKØ3 DD UNIT=339Ø,VOL=SER=&serv3,
)ENDSEL
//    DSN=&user..&hvdb..&hvts..WK3.&dsufb,
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,DELETE),
//    DCB=(BUFNO=1Ø)
)SEL &vol3 EQ Ø
//SORTWKØ4 DD UNIT=SYSDA,
)ENDSEL
)SEL &vol3 EQ 1
//SORTWKØ4 DD UNIT=339Ø,VOL=SER=&serv3,
)ENDSEL
//    DSN=&user..&hvdb..&hvts..WK4.&dsufb,
//    SPACE=(TRK,(&pri,&sec,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,DELETE),
```

```
//    DCB=(BUFNO=1Ø)
//SYSIN    DD  *
  REORG TABLESPACE &hvdb..&hvts
                   LOG YES
        COPYDDN (SYSCOPY)
        KEEPDICTIONARY
)SEL &rst EQ YES
  RUNSTATS TABLESPACE &hvdb..&hvts
               INDEX (ALL)
            SHRLEVEL REFERENCE
)ENDSEL
//*
)SEL &trg EQ YES
//*---- CREATE TRIGGERS -----------------------------
//RUNSQL2 EXEC PGM=IKJEFTØ1
//STEPLIB DD DISP=SHR,DSN=DSN71Ø.SDSNLOAD
//        DD DISP=SHR,DSN=CEE.SCEERUN
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(&db2)
 RUN PROGRAM(DSNTIAD) PLAN(DSNTIA71) -
     LIB('DSN71Ø.RUNLIB.LOAD.DSNN')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD *
  CREATE TRIGGER T&hvtg.I AFTER INSERT
  ON &cre..&tab REFERENCING NEW AS N
  FOR EACH ROW MODE DB2SQL
  INSERT INTO INFO.LOGI (CRE, TAB, KEY, AKC)
  VALUES ('&cre', '&tab', N.A&tid._ID, 'I');
)BLANK 1
  COMMIT;
  GRANT EXECUTE
        ON PACKAGE T&hvtg.I.* TO PUBLIC;
)BLANK 1
  CREATE TRIGGER T&hvtg.U AFTER UPDATE
  ON &cre..&tab REFERENCING NEW AS N
  FOR EACH ROW MODE DB2SQL
  INSERT INTO INFO.LOGI (CRE, TAB, KEY, AKC)
  VALUES ('&cre', '&tab', N.A&tid._ID, 'U');
)BLANK 1
  COMMIT;
  GRANT EXECUTE
        ON PACKAGE T&hvtg.U.* TO PUBLIC;
)BLANK 1
  CREATE TRIGGER T&hvtg.D AFTER DELETE
  ON &cre..&tab REFERENCING OLD AS O
  FOR EACH ROW MODE DB2SQL
  INSERT INTO INFO.LOGI (CRE, TAB, KEY, AKC)
  VALUES ('&cre', '&tab', O.A&tid._ID, 'D');
```

```
)BLANK 1
  COMMIT;
  GRANT EXECUTE
        ON PACKAGE T&hvtg.D.* TO PUBLIC;
/*
)ENDSEL
```

For the first solution, the REXX procedure ICOL generates the following sample JCL on the DSN8710.EMP table:

```
//SYSADMX JOB (777-ICOL),'ICOL',
//             NOTIFY=SYSADM,REGION=4M,
//             CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//RUNSQL1 EXEC PGM=IKJEFT01
//STEPLIB DD DISP=SHR,DSN=DSN710.SDSNLOAD
//        DD DISP=SHR,DSN=CEE.SCEERUN
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(DSNN)
 RUN PROGRAM(DSNTIAD) PLAN(DSNTIA71) -
     LIB('DSN710.RUNLIB.LOAD.DSNN')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD *
  ALTER TABLE DSN8710.EMP ADD ACOLUMN_ID INTEGER
                    GENERATED ALWAYS AS IDENTITY
                    (START WITH 1,
                     INCREMENT BY 1,
                     CACHE 20, NO CYCLE);
  COMMENT ON COLUMN DSN8710.EMP.ACOLUMN_ID IS 'IDENTITY KOLONA';
  COMMIT;
  CREATE UNIQUE INDEX INFO.X418973
         ON DSN8710.EMP
          ( ACOLUMN_ID DESC )
         USING STOGROUP GLLM06
                PRIQTY 100
                SECQTY 100
               FREEPAGE 5
                PCTFREE 10
         BUFFERPOOL BP3 ;
//*---- TERMINATE UTILITY ----------------------------
//TERMU   EXEC PGM=IKJEFT01,COND=(4,LT)
//STEPLIB  DD DSN=DSN710.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   DSN SYSTEM(DSNN)
   -TERM UTILITY(SYSADM.REORC)
   END
/*
//*---- REORG DSN8D71A.DSN8S71E
```

```
//REOR1  EXEC DSNUPROC,SYSTEM=DSNN,REGION=4Ø96K,
//      UID='SYSADM.REORC',UTPROC=''
//STEPLIB DD DSN=DSN71Ø.SDSNLOAD,DISP=SHR
//SYSREC DD UNIT=SYSDA,
//    DSN=SYSADM.DSN8D71A.DSN8S71E.REC.DØ68Ø752,
//    SPACE=(TRK,(14,4,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,CATLG)
//SORTOUT DD UNIT=SYSDA,
//    DSN=SYSADM.DSN8D71A.DSN8S71E.OUT.DØ68Ø752,
//    SPACE=(TRK,(14,4,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,CATLG)
//SYSUT1 DD UNIT=SYSDA,
//    DSN=SYSADM.DSN8D71A.DSN8S71E.UT1.DØ68Ø752,
//    SPACE=(TRK,(14,4,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,CATLG)
//SYSCOPY DD UNIT=SYSDA,
//    DSN=SYSADM.DSN8D71A.DSN8S71E.SYSCOPY.DØ68Ø752,
//    SPACE=(TRK,(14,4,),RLSE,,ROUND),
//    DISP=(MOD,CATLG,CATLG)
//SORTWKØ1 DD UNIT=SYSDA,
//    DSN=SYSADM.DSN8D71A.DSN8S71E.WK1.DØ68Ø752,
//    SPACE=(TRK,(14,4,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,DELETE),
//    DCB=(BUFNO=1Ø)
//SORTWKØ2 DD UNIT=SYSDA,
//    DSN=SYSADM.DSN8D71A.DSN8S71E.WK2.DØ68Ø752,
//    SPACE=(TRK,(14,4,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,DELETE),
//    DCB=(BUFNO=1Ø)
//SORTWKØ3 DD UNIT=SYSDA,
//    DSN=SYSADM.DSN8D71A.DSN8S71E.WK3.DØ68Ø752,
//    SPACE=(TRK,(14,4,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,DELETE),
//    DCB=(BUFNO=1Ø)
//SORTWKØ4 DD UNIT=SYSDA,
//    DSN=SYSADM.DSN8D71A.DSN8S71E.WK4.DØ68Ø752,
//    SPACE=(TRK,(14,4,),RLSE,,ROUND),
//    DISP=(NEW,DELETE,DELETE),
//    DCB=(BUFNO=1Ø)
//SYSIN    DD  *
  REORG TABLESPACE DSN8D71A.DSN8S71E
                   LOG YES
        COPYDDN (SYSCOPY)
        KEEPDICTIONARY
  RUNSTATS TABLESPACE DSN8D71A.DSN8S71E
               INDEX (ALL)
            SHRLEVEL REFERENCE
//*
//*---- CREATE TRIGGERS -----------------------------
//RUNSQL2 EXEC PGM=IKJEFTØ1
```

```
//STEPLIB DD DISP=SHR,DSN=DSN71Ø.SDSNLOAD
//        DD DISP=SHR,DSN=CEE.SCEERUN
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(DSNN)
 RUN PROGRAM(DSNTIAD) PLAN(DSNTIA71) -
     LIB('DSN71Ø.RUNLIB.LOAD.DSNN')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD *
  CREATE TRIGGER T418973I AFTER INSERT
  ON DSN871Ø.EMP REFERENCING NEW AS N
  FOR EACH ROW MODE DB2SQL
  INSERT INTO INFO.LOGI (CRE, TAB, KEY, AKC)
  VALUES ('DSN871Ø', 'EMP', N.ACOLUMN_ID, 'I');
  COMMIT;
  GRANT EXECUTE
        ON PACKAGE T418973I.* TO PUBLIC;
  CREATE TRIGGER T418973U AFTER UPDATE
  ON DSN871Ø.EMP REFERENCING NEW AS N
  FOR EACH ROW MODE DB2SQL
  INSERT INTO INFO.LOGI (CRE, TAB, KEY, AKC)
  VALUES ('DSN871Ø', 'EMP', N.ACOLUMN_ID, 'U');
  COMMIT;
  GRANT EXECUTE
        ON PACKAGE T418973U.* TO PUBLIC;
  CREATE TRIGGER T418973D AFTER DELETE
  ON DSN871Ø.EMP REFERENCING OLD AS O
  FOR EACH ROW MODE DB2SQL
  INSERT INTO INFO.LOGI (CRE, TAB, KEY, AKC)
  VALUES ('DSN871Ø', 'EMP', O.ACOLUMN_ID, 'D');
  COMMIT;
  GRANT EXECUTE
        ON PACKAGE T418973D.* TO PUBLIC;
/*
```

## THE SECOND SOLUTION

Each table that is a candidate for propagation is altered with a timestamp column. This column will be used later in a trigger definition.

Here is an example using installation sample table DSN8710.EMP:

```
ALTER TABLE DSN871Ø.EMP ADD ATR_TIMES TIMESTAMP
            NOT NULL WITH DEFAULT
```

The column ATR_TIMES is changed from the default value only

by an INSERT or LOAD operation. If you want to have the correct timestamp value for an UPDATE operation, define the update trigger.

You can propagate changed data to the local server in a specific date range. Example:

```
SELECT *
FROM DSN8710.EMP
WHERE DATE(ATR_TIMES) <= CURRENT DATE – 2 DAYS
WITH UR
```

The query will catch data modified in the last three days.

## TCOL – REXX DRIVER PROCEDURE

```
/* REXX */
/* TRACE R */
/* YOU MUST ALLOCATE PDS FILE IN YOUR ENVIRONMENT         */
"ALLOC DD(DD1) DSN('SKUPNI.CNTL("USERID()")') F(GRI) SHR REUSE"
 /* DSNREXX Language Support                              */
 Address TSO "SUBCOM DSNREXX"
 IF RC THEN
 S_RC = RXSUBCOM(ADD,DSNREXX,DSNREXX)
 Top:
 address ispexec "display panel(TCOLM)"
 if rc=8 then Exit
 tabela=tab
 SSID = db2
 ADDRESS DSNREXX "CONNECT" SSID
 SQLSTMT="SELECT 1, SUBSTR(CHAR(CURRENT TIMESTAMP),21,6)            ",
 " FROM SYSIBM.SYSTABLES                                            ",
 " WHERE CREATOR='"cre"'                                            ",
 "   AND NAME='"tabela"'                                            ",
 " WITH UR                                                          "
 Address DSNREXX "EXECSQL DECLARE C1 CURSOR FOR S1"
 Address DSNREXX 'EXECSQL PREPARE S1 FROM :SQLSTMT'
 Address DSNREXX "EXECSQL OPEN C1"
 Address DSNREXX "EXECSQL FETCH C1 INTO :HVCN, :HVTG"
 if sqlcode=100 then do
    zedsmsg = "Table not found"
    zedlmsg = "Table not found. Enter table name"
    Address DSNREXX "EXECSQL CLOSE C1"
    address ispexec "setmsg msg(isrz001)"
    signal top
 end
 KOLONA='ATR_TIMES'
 Address DSNREXX "EXECSQL CLOSE C1"
```

```
ROW.1='//'||userid()||'X JOB (777-TCOL),CLASS=A,                    '
ROW.2='//            MSGCLASS=X,NOTIFY='||userid()||',             '
ROW.3='//            MSGLEVEL=(1,1),USER=,REGION=4M                '
ROW.4='//*-----------------------------------------               '
ROW.5='//RUNSQL  EXEC PGM=IKJEFT01                                 '
ROW.6='//STEPLIB DD DISP=SHR,DSN=DSN710.SDSNLOAD                   '
ROW.7='//        DD DISP=SHR,DSN=CEE.SCEERUN                       '
ROW.8='//SYSTSPRT DD SYSOUT=*                                      '
ROW.9='//SYSTSIN  DD *                                             '
ROW.10=' DSN SYSTEM('||db2||')                                      '
ROW.11=' RUN PROGRAM(DSNTIAD) PLAN(DSNTIA71) -                      '
ROW.12='     LIB(''DSN710.RUNLIB.LOAD.DSNN'')                       '
ROW.13='//SYSPRINT DD SYSOUT=*                                     '
ROW.14='//SYSUDUMP DD SYSOUT=*                                     '
ROW.15='//SYSIN    DD *                                            '
ROW.16='ALTER TABLE '||cre||'.'||tabela||' ADD '||kolona||' TIMESTAMP'
ROW.17='            NOT NULL WITH DEFAULT;'
ROW.18='COMMENT ON COLUMN '||cre||'.'||tabela||'.'||kolona||' IS '
ROW.19='          ''SISTEMSKI DATUM SPREMEMBE'';'
ROW.20='COMMIT;'
ROW.21='CREATE INDEX '||cre||'.X'||hvtg
ROW.22='       ON '||cre||'.'||tabela
ROW.23='        ( '||kolona||' DESC )'
ROW.24='       USING STOGROUP GLLM06'
ROW.25='               PRIQTY 100'
ROW.26='               SECQTY 100'
ROW.27='             FREEPAGE 5'
ROW.28='              PCTFREE 10'
ROW.29='       BUFFERPOOL BP3 ;'
ROW.30='//*---- RUNSTATS INDEX ----------------------------      '
ROW.31='//IRUNST EXEC DSNUPROC,SYSTEM='||db2||',                    '
ROW.32='//      UID='''||userid()||'.IRUNST'',UTPROC=''''           '
ROW.33='//STEPLIB  DD DSN=DSN710.SDSNLOAD,DISP=SHR                 '
ROW.34='//SYSIN    DD  *                                           '
ROW.35='RUNSTATS INDEX ('||cre||'.X'||hvtg||')'
ROW.36='               REPORT NO                                   '
ROW.37='               UPDATE ALL                                  '
ROW.38='/*                                                         '
ROW.39='//RUNSQL  EXEC PGM=IKJEFT01                                '
ROW.40='//STEPLIB DD DISP=SHR,DSN=DSN710.SDSNLOAD                  '
ROW.41='//        DD DISP=SHR,DSN=CEE.SCEERUN                      '
ROW.42='//SYSTSPRT DD SYSOUT=*                                     '
ROW.43='//SYSTSIN  DD *                                            '
ROW.44=' DSN SYSTEM('||db2||')                                     '
ROW.45=' RUN PROGRAM(DSNTIAD) PLAN(DSNTIA71) -                     '
ROW.46='     LIB(''DSN710.RUNLIB.LOAD.DSNN'')                      '
ROW.47='//SYSPRINT DD SYSOUT=*                                     '
ROW.48='//SYSUDUMP DD SYSOUT=*                                     '
ROW.49='//SYSIN    DD *                                            '
ROW.50='CREATE TRIGGER '||cre||'.T'||hvtg||,
```

```
       ' AFTER UPDATE OF'
R=51
SQLSTMT="SELECT STRIP(NAME)                                                ",
" FROM SYSIBM.SYSCOLUMNS                                                   ",
" WHERE TBCREATOR='"cre"'                                                  ",
"   AND TBNAME='"tabelA"'                                                  ",
"   AND NAME<>'"kolona"'                                                   ",
" ORDER BY COLNO                                                           ",
" WITH UR                                                                  "
Address DSNREXX "EXECSQL DECLARE C1 CURSOR FOR S1"
Address DSNREXX 'EXECSQL PREPARE S1 FROM :SQLSTMT'
Address DSNREXX "EXECSQL OPEN C1"
Address DSNREXX "EXECSQL FETCH C1 INTO :HVCN"
do while(sqlcode=0)
   ROW.R='       '||HVCN
   Address DSNREXX "EXECSQL FETCH C1 INTO :HVCN"
   if sqlcode=0 then ROW.R=ROW.R||', '||HVCN
   Address DSNREXX "EXECSQL FETCH C1 INTO :HVCN"
   if sqlcode=0 then ROW.R=ROW.R||', '||HVCN
   Address DSNREXX "EXECSQL FETCH C1 INTO :HVCN"
   if sqlcode=0 then ROW.R=ROW.R||','
   R=R+1
end
Address DSNREXX "EXECSQL CLOSE C1"
ROW.R='       ON '||cre||'.'||TABELA||' REFERENCING NEW AS N        '
R=R+1
ROW.R='       FOR EACH ROW MODE DB2SQL                               '
R=R+1
ROW.R='       UPDATE '||cre||'.'||TABELA
R=R+1
ROW.R='       SET '||kolona||'=CURRENT TIMESTAMP             '
R=R+1
SQLSTMT="SELECT CASE(K.COLSEQ)                                             ",
"           WHEN 1 THEN '       WHERE  '||STRIP(K.COLNAME)         ",
"           ELSE '         AND  '||STRIP(K.COLNAME)                ",
"          END CONCAT '=N.'||STRIP(K.COLNAME)                      ",
"         , K.COLSEQ                                               ",
" FROM SYSIBM.SYSINDEXES I,                                        ",
"      SYSIBM.SYSKEYS K                                            ",
" WHERE TBCREATOR='"cre"'                                          ",
"   AND TBNAME='"TABELA"'                                          ",
"   AND UNIQUERULE<>'D'                                            ",
"   AND I.CREATOR=K.IXCREATOR                                      ",
"   AND I.NAME=K.IXNAME                                            ",
"   AND CREATOR||NAME=(SELECT MAX(CREATOR||NAME)                   ",
"                     FROM SYSIBM.SYSINDEXES                       ",
"                     WHERE TBCREATOR='"cre"'                      ",
"                       AND TBNAME='"TABELA"'                      ",
"                       AND UNIQUERULE<>'D')                       ",
" ORDER BY K.COLSEQ                                                ",
```

```
" WITH UR                                                                "
Address DSNREXX "EXECSQL DECLARE C1 CURSOR FOR S1"
Address DSNREXX 'EXECSQL PREPARE S1 FROM :SQLSTMT'
Address DSNREXX "EXECSQL OPEN C1"
Address DSNREXX "EXECSQL FETCH C1 INTO :HVCN, :HVCS"
do while(sqlcode=Ø)
   ROW.R=HVCN
   Address DSNREXX "EXECSQL FETCH C1 INTO :HVCN, :HVCS"
   R=R+1
end
Address DSNREXX "EXECSQL CLOSE C1"
ROW.R=';'
R=R+1
ROW.R='COMMIT;'
R=R+1
ROW.R='GRANT EXECUTE ON PACKAGE '||'T'||hvtg||'.* TO PUBLIC;'
"EXECIO * DISKW GRI (STEM ROW. FINIS"
 ADDRESS ISPEXEC "EDIT DATASET('SKUPNI.CNTL("USERID()")')"
"EXECIO Ø DISKR GRI (FINIS"
ADDRESS TSO "FREE F(GRI)"
EXIT
```

## TCOLM – ENTRY PANEL

```
)ATTR
   $ type(text)   color(white) caps (off) hilite(reverse) intens(high)
   | type(text)   color(white) hilite(reverse) intens(high)
   ( type(text)   color(yellow)    hilite(reverse) intens(high)
   ) type(text)   color(green)                     intens(high)
   _ type(input)  color(red)       intens(high) pad(_)
)BODY WINDOW(36,12)
+
$    Alter timestamp column
| +                                |
| )  Db2     :_db2 +               |
| )  Creator :_cre     +           |
| )  Table   :_tab                +|
| +                                |
( Enter:Continue            PF3:End
)INIT
  if (&db2 ¬= ' ')
      .attr (db2) = 'pad(nulls)'
  if (&cre ¬= ' ')
      .attr (cre) = 'pad(nulls)'
  if (&tab ¬= ' ')
      .attr (tab) = 'pad(nulls)'
)PROC
  IF (.PFKEY = PFØ3) &PF3 = EXIT
  VPUT (TAB CRE DB2) PROFILE
)END
```

For the second solution, the REXX procedure TCOL generates the following sample JCL on the DSN8710.EMP table:

```
//SYSADMX JOB (777-TCOL),CLASS=A,
//            MSGCLASS=X,NOTIFY=SYSADM,
//            MSGLEVEL=(1,1),USER=,REGION=4M
//*--------------------------------------
//RUNSQL  EXEC PGM=IKJEFT01
//STEPLIB DD DISP=SHR,DSN=DSN710.SDSNLOAD
//        DD DISP=SHR,DSN=CEE.SCEERUN
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(DSNN)
 RUN PROGRAM(DSNTIAD) PLAN(DSNTIA71) -
     LIB('DSN710.RUNLIB.LOAD.DSNN')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD *
ALTER TABLE DSN8710.EMP ADD ATR_TIMES TIMESTAMP
            NOT NULL WITH DEFAULT;
COMMENT ON COLUMN DSN8710.EMP.ATR_TIMES IS
          'SISTEMSKI DATUM SPREMEMBE';
COMMIT;
CREATE INDEX DSN8710.X730639
       ON DSN8710.EMP
        ( ATR_TIMES DESC )
       USING STOGROUP GLLM06
               PRIQTY 100
               SECQTY 100
             FREEPAGE 5
              PCTFREE 10
       BUFFERPOOL BP3 ;
//*---- RUNSTATS INDEX -----------------------------
//IRUNST EXEC DSNUPROC,SYSTEM=DSNN,
//      UID='SYSADM.IRUNST',UTPROC=''
//STEPLIB  DD DSN=DSN710.SDSNLOAD,DISP=SHR
//SYSIN    DD  *
RUNSTATS INDEX (DSN8710.X730639)
               REPORT NO
               UPDATE ALL
/*
//RUNSQL  EXEC PGM=IKJEFT01
//STEPLIB DD DISP=SHR,DSN=DSN710.SDSNLOAD
//        DD DISP=SHR,DSN=CEE.SCEERUN
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(DSNN)
 RUN PROGRAM(DSNTIAD) PLAN(DSNTIA71) -
     LIB('DSN710.RUNLIB.LOAD.DSNN')
//SYSPRINT DD SYSOUT=*
```

```
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD *
CREATE TRIGGER DSN871Ø.T73Ø639 AFTER UPDATE OF
       EMPNO, FIRSTNME, MIDINIT,
       LASTNAME, WORKDEPT, PHONENO,
       HIREDATE, JOB, EDLEVEL,
       SEX, BIRTHDATE, SALARY,
       BONUS, COMM
       ON DSN871Ø.EMP REFERENCING NEW AS N
       FOR EACH ROW MODE DB2SQL
       UPDATE DSN871Ø.EMP
       SET ATR_TIMES=CURRENT TIMESTAMP
       WHERE  EMPNO=N.EMPNO
;
COMMIT;
GRANT EXECUTE ON PACKAGE T73Ø639.* TO PUBLIC;
```

*Bernard Zver*
*DBA*
*Informatika (Slovenia)*

BMC has announced second-generation SmartDBA DB2 Version 7 management tools for mainframe and distributed environments, with five mainframe DB2 and two distributed systems DB2 UDB applications. The tools automate database management tasks by providing tighter levels of integration and have new built-in intelligence features, said to help improve application uptime.

The five new DB2 tools span performance, administration, and recovery. System Performance for DB2 2.0 provides new navigation components that offer a task-oriented approach to system performance.

The new reporting function provides system health indicators and accounting and audit data without requiring the use of system management facility (SMF) and its associated overhead. Also, new reporting capabilities spot trends or real-time bottlenecks that can degrade DB2 subsystem and application performance if left unchecked.

Application Performance for DB2 2.0 helps improves application response times and overall DB2 performance by enabling users to analyse their DB2 index objects to improve performance.

For further information contact:
Syncsort, 50 Tice Boulevard, Woodcliff Lake, NJ 0767, USA.
Tel: (201) 93 8200.
URL: http://www.bmc.com/solutions/database.

* * *

NEON Systems has announced that its Shadow JDBC Adapter for mainframe integration has passed the WebSphere Self-Testing process and will be added to IBM's Self-Tested Software support page. The IBM-sponsored programme facilitates self-testing of WebSphere complementary software through an IBM-endorsed testing process.

Shadow software can be deployed with WebSphere to provide JCA or JDBC access to mainframe data sources and transaction environments, supporting DB2, CICS/TS, IMS/TM, IMS/DB, VSAM, ADABAS, Natural/ACI, flat files, IDMS, and other z/OS mainframe data and transactional sources.

For further information contact:
NEON Systems, 14100 Southwest Freeway, Suite 500, Sugarland, TX 77478, USA.
Tel: (281) 491 4200.
URL: http://www.neonsys.com.

* * *

IBM has enhanced five DB2 for z/OS tools. Administration tools and utilities include the enhanced DB2 Automation Tool for z/OS, V1.3, which streamlines database management tasks. Performance management tools include DB2 Buffer Pool Analyzer for z/OS, V1.2 and DB2 Performance Monitor for z/OS, V7.2.

The recovery and replication tools are represented by DB2 Log Analysis Tool for z/OS, V1.3 and DB2 Object Restore for z/OS, V1.3, with additional back-up and recovery functions.

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

* * *