# 158

# DB2 update

*December 2005*

## In this issue





A xephon PUBLICATION

## Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

## Editor

Trevor Eddolls
E-mail: trevore@xephon.com

## Publisher

Colin Smith
E-mail: info@xephon.com

## Subscriptions and back-issues

A year's subscription to *DB2 Update*, comprising twelve monthly issues, costs $380.00 in the USA and Canada; £255.00 in the UK; £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 2001 issue, are available separately to subscribers for $33.75 (£22.50) 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.



## Contributions

When Xephon is given copyright, articles published in *DB2 Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.



*Printed in England.*

# DB2 LUW – new list history table function

The **db2 list history** command has been around for a long time. It is a very useful command that tells you when back-ups, reorgs, etc were taken, when table spaces were created (and dropped), and lots more – but it can produce a huge amount of output! With DB2 UDB V8 FP9 you can now use a table function called **ADMIN_LIST_HIST()** to query the database history file using SQL.

If you are not connected to a database when you issue the command, a connection will be made for you to the database that is defined in the DB2DBDFT registry variable. If a database is not defined and you are not connected, you will receive an SQL1328N error message if you try to execute the SQL.

The SQL in this article was run on a Windows 2000 Professional system running DB2 UDB 8.2. FP10 for LUW.

We can see all the available columns in the table function by issuing the command:

```
>db2 describe select * from table(sysproc.ADMIN_LIST_HIST()) as s.
```

The columns available are: BACKUP_ID(VARCHAR), CMD_TEXT(CLOB), COMMENT(VARCHAR), DBPARTITIONNUM(SMALLINT), DEVICETYPE(CHAR), EID(BIGINT), END_TIME(VARCHAR), ENTRY_STATUS(CHAR), FIRSTLOG(VARCHAR), LASTLOG(VARCHAR), LOCATION(VARCHAR), NUM_TBSPS(INT), OBJECTTYPE(CHAR), OPERATION(CHAR), OPERATIONTYPE(CHAR), SEQNUM(SMALLINT), SQLCABC(INT), SQLCAID(VARCHAR), SQLCODE(INT), SQLERRD1(INT), SQLERRD2(INT), SQLERRD3(INT), SQLERRD4(INT), SQLERRD5(INT), SQLERRD6(INT), SQLERRMC(VARCHAR), SQLERRML(SMALLINT), SQLERRP(VARCHAR), SQLSTATE(VARCHAR), SQLWARN(VARCHAR), START_TIME(VARCHAR),

TABNAME(VARCHAR), TABSCHEMA(VARCHAR), and TBSPNAMES(CLOB).

There are a lot of columns, so let's use the table function and highlight one or two of the columns. How many full off line back-ups of my database have I taken? First let me use the list history command.

```
>db2 list history backup all for sample

                    List History File for sample

Number of matching file entries = 1


 Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log  Backup ID
 -- --- ------------------ ---- --- ------------ ------------ ----------
  B  D  20050816194616001   F    D  S0000000.LOG S0000000.LOG
 ----------------------------------------------------------------------
  Contains 2 tablespace(s):

  00001 SYSCATSPACE
  00002 USERSPACE1
 ----------------------------------------------------------------------
    Comment: DB2 BACKUP SAMPLE OFFLINE
 Start Time: 20050816194616
   End Time: 20050816194637
     Status: A
 ----------------------------------------------------------------------
  EID: 1 Location:
c:\db2_backups\SAMPLE.0\DB2\NODE0000\CATN0000\20050816
```

This is the output for just one entry – if you have five or six back-ups registered, you really need a program to pick out the information that you want. The output shows the operation performed as well as the start/end times (but not the duration) and the location of the back-up.

I could run the following query using the table function to give me the same information:

```
>db2 select substr(operation,1,1) as "OP", substr(operationtype,1,1) as
"OT", substr(start_time,1,14) as "Start_T", substr(end_time,1,14) as
"End_T", integer(timestamp(end_time)-timestamp(start_time)) as
"Duration_in_secs", substr(location,1,60) as "Location" from
table(sysproc.ADMIN_LIST_HIST()) as s where operation = 'B'
```

```
OP OT START_T        END_T          DURATION_IN_SECS LOCATION
-- -- -------------- -------------- ---------------- ------------------
---------------------------------------
B  F  20050816194616 20050816194637               21
c:\db2_backups\SAMPLE.0\DB2\NODE0000\CATN0000\20050816
```

This query gives me the duration of the back-up, which I can store and then review to see whether the back-up times are increasing over time.

This is just one example. The Information Center gives a list of all the operations that are recorded in the history file. I thought it would be useful to list them here:

A – Create table space, B – Backup, C – Load copy, D – Dropped table, F – Roll forward, G – Reorganize table, L – Load, N – Rename table space, O – Drop table space, Q – Quiesce, R – Restore, T – Alter table space, U – Unload, X – Archive log.

Here are some more examples. Suppose we want to find out when a table space was created. If I create a table space using the following SQL:

```
CREATE REGULAR TABLESPACE TABSP1 PAGESIZE 4 K MANAGED BY SYSTEM USING
('C:\temp\cont1' ) EXTENTSIZE 16 OVERHEAD 10.5 PREFETCHSIZE 16
TRANSFERRATE 0.14 BUFFERPOOL  IBMDEFAULTBP  DROPPED TABLE RECOVERY OFF;
```

The query I would run is:

```
>db2 select substr(operation,1,1) as "OP", substr(start_time,1,14) as
"Start_T", substr(tbspnames,1,20) as "Tablespace_name", sqlcode from
table(sysproc.ADMIN_LIST_HIST()) as s where operation = 'A'

OP START_T        TABLESPACE_NAME      SQLCODE
-- -------------- -------------------- -----------
A  20050913141522 'SYSTOOLSPACE'                -
A  20050923090725 'TABSP1'                   -298
A  20050923090800 'TABSP1'                      -
```

I have included the SQLCODE column because we can then see whether the create operation completed successfully or not. You can see in the above example that a create tablespace operation failed with a SQL0298N error before it was successfully executed.

Another example is to check whether a load operation has been performed on any table in the database. This is especially important if you are using HADR. Suppose the LOAD command used was:

```
LOAD FROM "c:\temp\loadf.txt" OF DEL METHOD P (1, 2) MESSAGES
"c:\temp\msg.txt" INSERT INTO DB2ADMIN.TBLOAD (ID1, ID2) COPY YES TO
"c:\temp" INDEXING MODE AUTOSELECT;
```

Then I could run the following query to check when/if a load has been run and the affected table.

```
>db2 select substr(operation,1,1) as "OP", substr(start_time,1,14) as
"Start_T", substr(tabschema,1,20) as "table_schema",
substr(tabname,1,20) as "Table_name", sqlcode from
table(sysproc.ADMIN_LIST_HIST()) as s where operation =  'L'

OP START_T        TABLE_SCHEMA         TABLE_NAME           SQLCODE
-- -------------- -------------------- -------------------- -----------
L  20050923092637 DB2ADMIN             TBLOAD                         -
```

If you now run the ‘backup’ query you will not see an entry for the **COPY YES** backup parameter you would have to query for the ‘C’ (**LOAD COPY**) operation. If I ran the query looking for the ‘C’ operation, I would also select the location, because in a HADR environment this would have to be shared between the primary and standby systems for the log to roll forward through the load to complete successfully.

```
>db2 select substr(operation,1,1) as "OP", substr(start_time,1,14) as
"Start_T", substr(tabschema,1,20) as "table_schema",
substr(tabname,1,20) as "Table_name", sqlcode, substr(location,1,60) as
"Location" from table(sysproc.ADMIN_LIST_HIST()) as s where operation =
'C'

OP START_T        TABLE_SCHEMA         TABLE_NAME           SQLCODE
LOCATION
-- -------------- -------------------- -------------------- ---------- -
----------------------------------------
C  20050923092637 DB2ADMIN  TBLOAD            -
c:\temp\SAMPLE.4\DB2\NODE0000\CATN0000\20050923\092637.001
```

If I want to keep track of any tables that have been dropped using the ‘D’ operation, the table space that contained the table needs to have been created with the **DROPPED TABLE RECOVERY** option enabled.

The EID column is the history file entry number that is useful if you want to see the order in which commands were executed and their start times, as shown below:

```
>db2 select eid, substr(operation,1,1) as "OP",substr(start_time,1,14)
as "Start_T" from table(sysproc.ADMIN_LIST_HIST()) as s

EID                  OP START_T
-------------------- -- --------------
                   1 B  20050816194616
                   2 A  20050913141522
                   3 B  20050923085856
                   4 A  20050923090725
                   5 A  20050923090800
                   6 L  20050923092637
                   7 C  20050923092637
```

You can tidy up the output using a **CASE** statement, but I hope you get the idea.

I hope I have shown how easy it is to use the list history table function. Because I can use SQL to query the history file it means I don't have to write and maintain a program, which I would otherwise have to do.

*C Leonard*
*Freelance Consultant (UK)*


## A path to DB2 lock information

A key method to tuning your IBM DB2 Universal Database for Linux, Unix, and Windows (DB2 UDB) servers is to understand any potential locking or statement blocking that could be going on in your system. In this article I want to give you 'Locking 101' (ie a beginner's guide) with respect to getting this type of information on your DB2 UDB servers to aid in performance tuning.

Essentially there are six methods by which you can retrieve locking information about your DB2 UDB server (the one you

select will depend on what you want to do and from where you want to do it). Specifically, you can get lock information from the following sources:

- Monitor switches
- GET SNAPSHOT FOR statement
- db2pd problem determination tool
- A set of Application Programming Interfaces (APIs)
- Snapshot monitor SQL table functions
- Activity Monitor.

## MONITOR SWITCHES

You can use monitor switches to instruct DB2 UDB to collect information about the locks on your system (in fact, this is really a starting point for some of the other methods). You can actually use monitor switches to monitor many more things, but that's beyond the scope of this article.

You can interact with these switches graphically via the Control Center, through API interfaces into them, or via the DB2 command line.

Figure 1 shows the status of the various monitor switches on

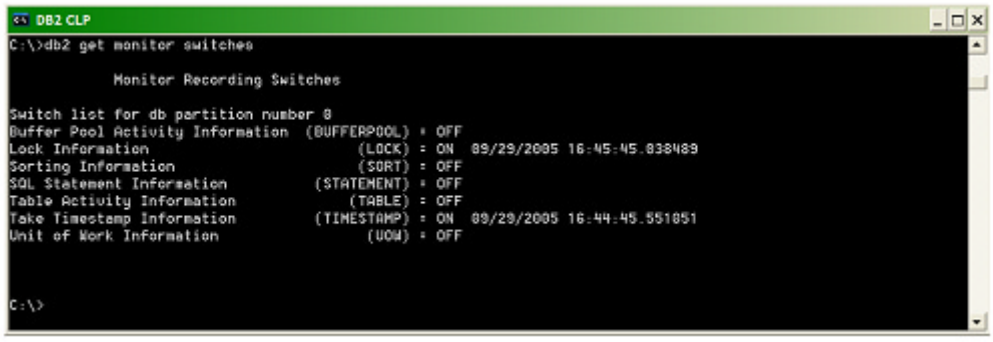


*Figure 1: Status of monitor switches*

my DB2 UDB server – you can see I've turned on the Lock Information monitor switch.

To get full lock monitor information, you need to set the LOCK monitor switch to ON (as shown in Figure 1). Keep in mind that you need to have one of the following authorities to change the status of a monitor switch: SYSADM, SYSCTRL, SYSMAINT, or SYSMON (SYSMON is a new authority specifically added in DB2 V8.2 to allow the assignment of lower level authorities for monitoring). To get the information from the switches, you can use a number of techniques.

## THE GET SNAPSHOT FOR STATEMENT

You can retrieve lock information from your DB2 UDB server (assuming it's being collected) by issuing the **GET SNAPSHOT** statement for all databases, a specific database, a specific application (identified through the application ID), and more.

Figure 2 shows output from one of the **GET SNAPSHOT** commands. Note that the database SMART currently has 32

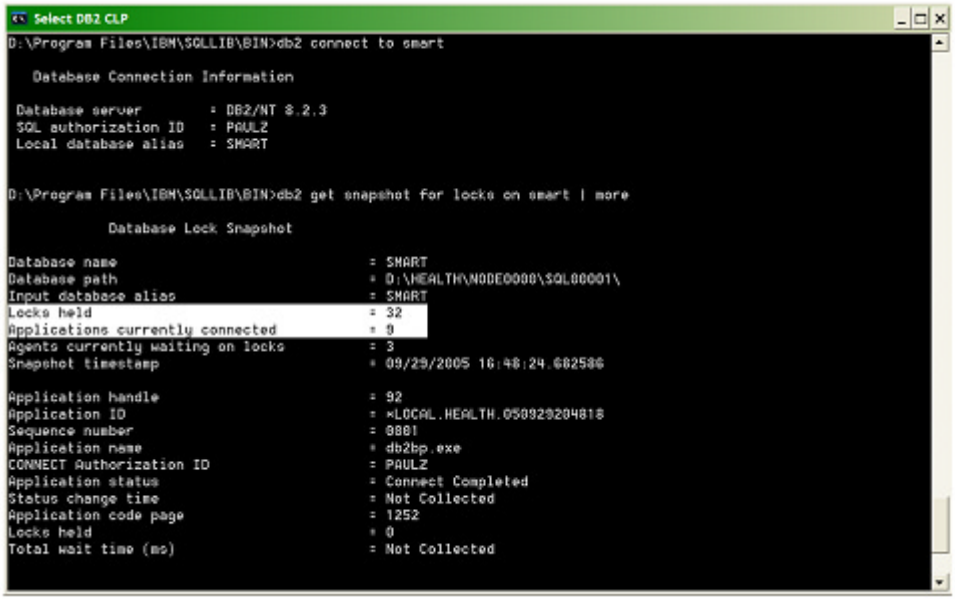


*Figure 2: GET SNAPSHOT output*

locks that have been generated from up to nine different applications accessing the database.

## THE DB2PD PROBLEM DETERMINATION TOOL

The db2pd tool first made its way into DB2 UDB in the Version 8.2 release. Those familiar with the Informix database product

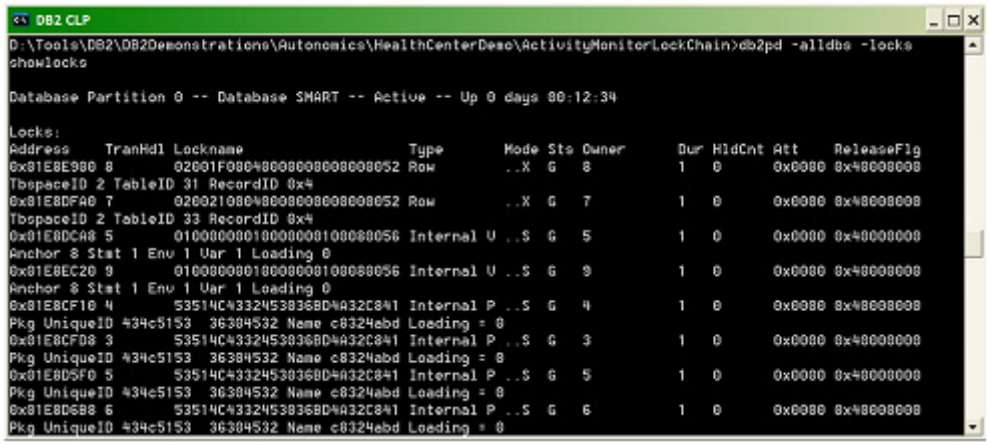


*Figure 3: Example db2pd command*

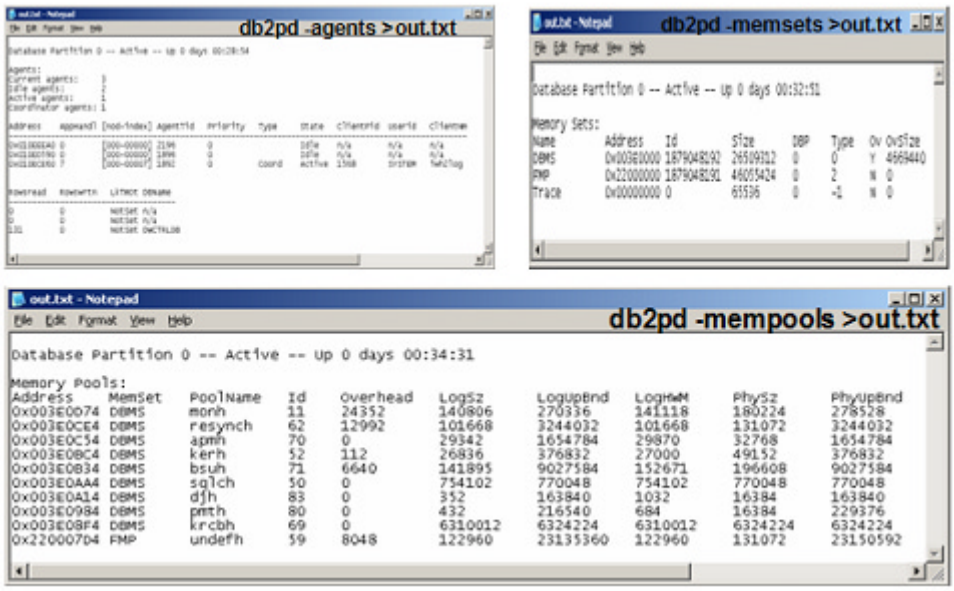


*Figure 4: Additional db2pd information*

love this tool because it provides a very rich method of retrieving all sorts of locking, latching, memory, etc, information in a non-obtrusive fashion, since it doesn't require latching. This by far is the most non-impact way to get these types of detail from your DB2 UDB server.

You can use the db2pd tool in a manner similar to the following:

```
     db2pd -db <database_name>
[-locks showlocks | locks wait | transactions | agents]
```

An example of using this command is shown in Figure 3.

The db2pd tool can collect more information than just locks. For example, you can get information on agents, memory sets, memory pools, and much more, as shown in Figure 4.

## A SET OF APPLICATION PROGRAMMING INTERFACES

DB2 UDB provides the ability to capture database snapshots programmatically using an API in your applications. For example, the APIs (db2MonitorSwitches, db2GetSnapshot, db2GetSnapshotSize, and db2ResetMonitor) can be used to work with snapshot data and function in the same manner as a DBA can work with them via the command line or Control Center.

Using APIs essentially leads you to the same results as any other method, only you are accessing them programmatically. The long-term direction of DB2 UDB is to move towards a Web service-fronted model, whereby a simple call to a Web service abstracts the details of the implemented API performing the function. You can learn all about using APIs to access locking information in the *DB2 Administrative Application Programming Interface Reference*.

## SNAPSHOT MONITOR SQL TABLE FUNCTION

Perhaps the easiest ways for DBAs to get locking information about their databases is via SQL (of course the graphical

method covered in the next section is even easier, but SQL calls allow DBAs to create scripts to get this information, which the experienced ones like).

Using the snapshot monitor SQL table function doesn't return all the information that the other methods do, but the simplicity of a SQL call is well liked by DBAs and the result set is mostly complete. In addition to this, since it is just a table function, you have the added flexibility of creating a view on these functions to persist this information.

The following SQL table functions are available for locking information:

- SNAPSHOT_LOCK
- SNAPSHOT_APPL
- SNAPSHOT_LOCKWAIT
- SELECT * FROM TABLE(SNAPSHOT_APPL ('<database_name>',-1)) AS SNAPSHOTAPPLOCKS

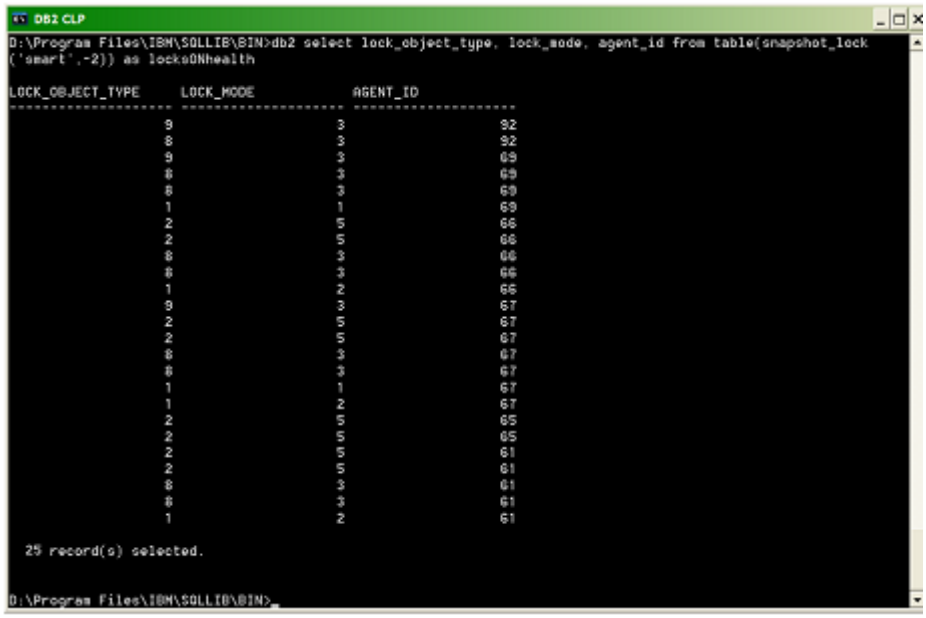


*Figure 5: Using SQL to retrieve locking information*

(There are many permutations to this one.)

Figure 5 shows locking information being retrieved using SQL on a DB2 UDB server.

## ACTIVITY MONITOR

The Activity Monitor can help you monitor application performance, application concurrency, resource consumption, and SQL statement usage. It can assist you in diagnosing

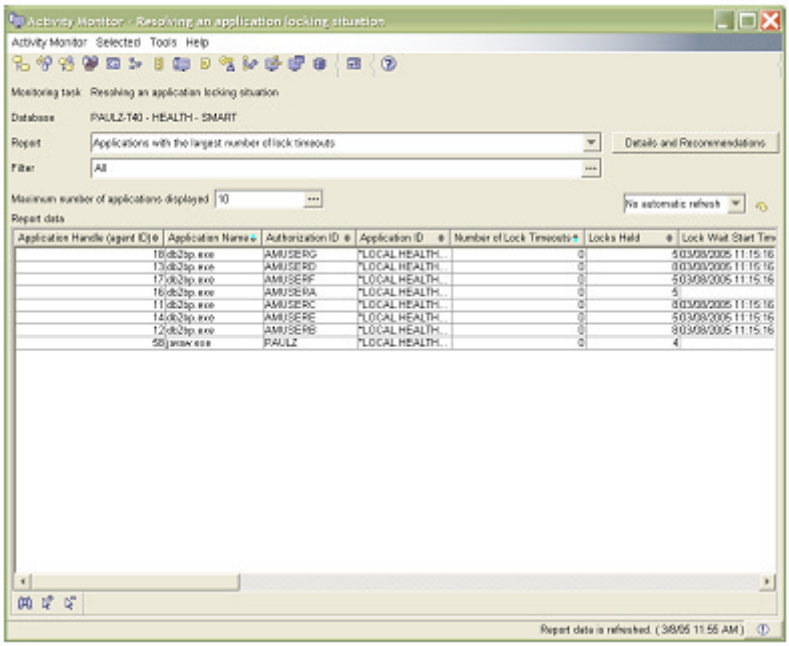


*Figure 6: Activity Monitor*

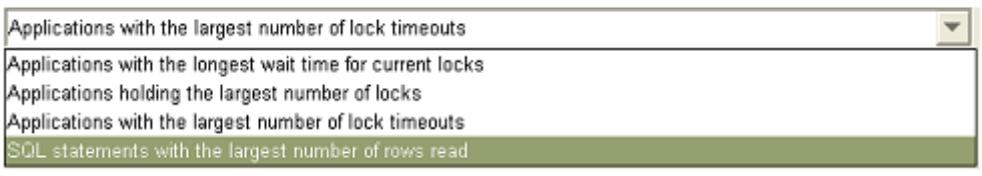


*Figure 7: Activity Monitor reports*

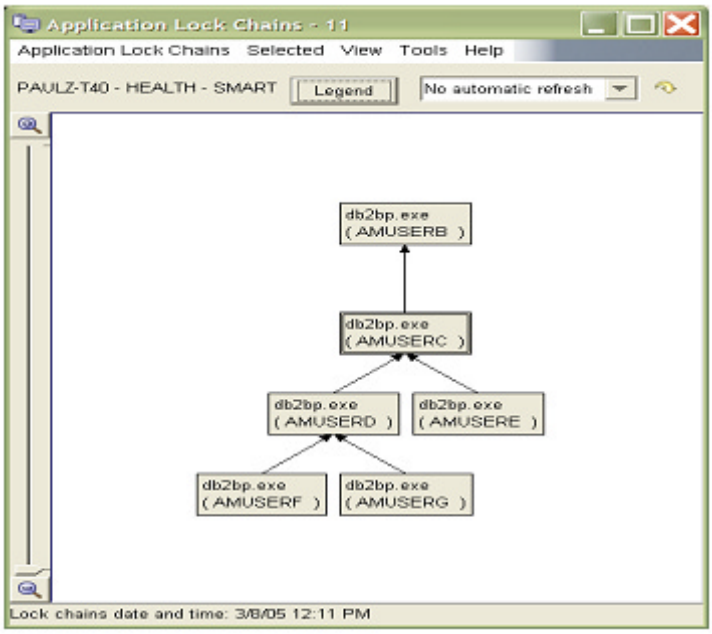


*Figure 8: Lock chains*

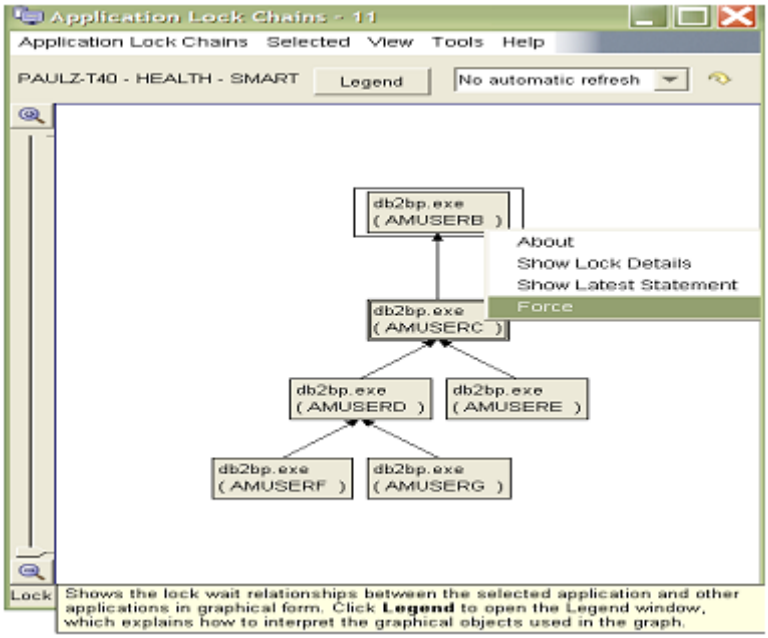


*Figure 9: Displaying lock details*

database performance problems such as lock-waiting situations, and in tuning queries for optimal utilization of the database resources.

You can see in Figure 6 that there are a number of locks waiting on other locks on my DB2 UDB server, but haven't timed out yet (note that you filter the results in this window as well).

The Activity Monitor comes with a number of reports that DB2 UDB automatically generates for you, as shown in Figure 7.

In Figure 8, you can see the lock chains associated with the authorization IDs. Note that AMUSERB's application is causing blocking on AMUSERC's application, which in turn is blocking AMUSERD, which in turn blocks two other applications.

If you right-click any of the applications connected to the database, you can force the lock, find out more details about the lock, or show the SQL statement that is causing the lock – see Figure 9.

## UNLOCKING THE MYSTERY

As you can see, DB2 UDB has many facilities to retrieve lock-based information about your DB2 UDB server. You'll end up choosing the right method and the right interface for whatever it is you're trying to do. For example, operational DBAs are likely to leverage the SQL table functions and the Activity Monitor, while partners writing applications that integrate into DB2 UDB will leverage the API. The good news is that DB2 UDB provides you with the flexibility you need for whatever method you want.

*Paul C Zikopoulos*
*Senior Specialist, Competitive Technology Team*
*IBM (Canada)*

## Can Oracle work with DB2?

Two of Oracle's recent acquisitions seem to have brought it closer to the world of DB2. The big question for Oracle is, what should it do next?

Following its acquisition of PeopleSoft, and more recently Retek and Siebel, Oracle now has a large number of DB2 users as customers. Oracle recently announced that it plans to work with IBM on compatibility between WebSphere and Oracle's Project Fusion, so it might support DB2 on its Fusion middleware. Certainly, at the Oracle Open World conference in San Francisco in September, Larry Ellison was not giving any hints about Oracle's future direction in terms of DB2.

Ellison spoke about talking to customers about migrating from DB2 to Oracle. He also addressed the rumour that Oracle would kill the DB2-based Siebel OnDemand CRM offering; he said that Oracle had no plans to kill it. However, they may have plans later – when they finally decide what to do with all their DB2 users.

In a compromise answer, Ellison was saying that Oracle would certify non-Oracle databases (ie DB2 for I-Flex and Retek), but needed more time to decide whether Fusion applications would support these non-Oracle databases. Will Oracle embrace DB2? We shall see!

*Nick Nourse*
*Independent Consultant (UK)*


## Managing DDL changes – part 2

*This month we continue publishing the REXX programs that enable you to manage source members related to DB2 DDL definitions.*

## CHGPOPUP

```
)ATTR
 [ TYPE(OUTPUT) INTENS(HIGH) CAPS(OFF) JUST(ASIS)
)BODY WINDOW(5Ø,3)
+
[text
)END

  VER(&CHGVTARG,LIST,T,R,A,P)
  VPUT (CHGVTARG) PROFILE
  &ZQ = TRUNC(&ZCMD,'.')
  &ZTRAIL = .TRAIL
)END
```

## CHGPMEMU

```
)attr
/*************************************************************************/
/*                                                                       */
/*************************************************************************/
~ type(input) intens(high) caps(on)  color(white)
% type(output) intens(high) caps(off) skip(on) color(green)
# type(text) intens(low) skip(on)
@ TYPE( INPUT) INTENS(LOW ) CAPS(On)  PADC('_') Color(green)
] TYPE( INPUT) INTENS(LOW ) CAPS(Off) PADC('_') color(green)
)body lmsg(msgl) smsg(msgs) window(65,19)
+               Member connection
+ %msgl%msgs++
+Membername          %mbrnm
+Current release name%oprojnm
+New release name    ~projnm  +
+Execute DDL     Y/N ~Z+
+DDL sequence    Y/N ~Z +
+
)init
  .HELP = CHGHMEMU
  .zvars = '(exein ddlseq)'
)reinit
)proc
    ver(&projnm,nb)
    ver(&exein,nb,list,Y,N)
    ver(&ddlseq,nb,num)
)end
```

## CHGPMEMB

```
)ATTR
/*************************************************************************/
```

```
/*                                                                    */
/**********************************************************************/
   # TYPE(OUTPUT) INTENS(HIGH) SKIP(ON) CAPS(OFF) COLOR(GREEN)
   ] TYPE( INPUT) INTENS(HIGH) CAPS(ON ) JUST(LEFT ) PAD('.')
   $ TYPE(OUTPUT) INTENS(LOW ) CAPS(Off) JUST(LEFT ) PADC('_')
COLOR(GREEN)
   + TYPE(  TEXT) INTENS(LOW ) SKIP(ON)
)BODY  EXPAND(~~)
%-~-~- Connected members -~-~-
%COMMAND ===>_PCMD                                ~ ~%SCROLL ===>_AMT +
%
%   Project       :#projnm  %
%   SD-release    :#sdrel
%   Status        :#status  %
%   Description   :#projtx
%
%                  New/ --DDL--          Connect    Connect  Stat
%   Mbrname   Type  Chg incl seq Userid   Date       Time     mbr
%   --------  ----  -- ---- -- ------  ---------- -------- ----
)MODEL ROWS(SCAN)
]Z $Z        $Z    $Z    $Z  $Z $Z       $Z         $Z       $Z
)INIT
 .ZVARS='(LCMD mbrnm mbrtype srctxt exein ddlseq userid upddt updtd +
          mbrstat)'
  &LCMD=&Z
  &PCMD=''
  &AMT = CSR
  .HELP = CHGHMEMB
  .CURSOR = LCMD
)REINIT
  IF (.MSG = ' ')
    &LCMD=' '
    REFRESH(LCMD)
)PROC
    VER(&LCMD,LIST,B,D,E,U)
)END
```

## CHGPOVER

```
)ATTR
/**********************************************************************/
/*                                                                    */
/*|        |    |    |                                               |*/
/**********************************************************************/
  + TYPE(TEXT)   INTENS(LOW) SKIP(ON)
  [ TYPE(OUTPUT) INTENS(HIGH) SKIP(ON) CAPS(OFF)
  @ TYPE(TEXT)   INTENS(LOW) SKIP(ON) COLOR(RED)
  # TYPE(TEXT)   INTENS(LOW) SKIP(ON) COLOR(RED) HILITE(BLINK)
)BODY EXPAND(~~)
```

```
%~-~- Promote DB2-release -~-~-
%OPTIE ===>_ZCMD                                                          +
+                                                    User   : &ZUSER +
+                                                    Date   : &ZDATE  +
+                                                    Time   : &ZTIME +   +
+                                                    System :@&chgvstat  +
+                                                                        +
+                                                                        +
+                                                                        +
+                   Release name : &projnm  +                            +
+                                                                        +
+                   Status       :_Z+                                    +
+                                                                        +
%               T +copy release from PROD to TEST                       +
%               R +recall release from ACPT to TEST                     +
%               A +move release to ACPT                                 +
%               P +move release to PROD                                 +
+                                                                       +
+[emsg                                                                  +
+               [txtØ                                                   +
+               [txt9                                                   +
  #&olbtxt                                                              +
)INIT
  .HELP = CHGHOVER
  .ZVARS ='(CHGVTARG)'
  VGET (CHGVTARG)
)REINIT
)PROC
```

## CHGPPRJU

```
)attr
/*************************************************************************/
/*                                                                       */
/*************************************************************************/
~ type(input) intens(high) caps(on)  color(white)
% type(output) intens(high) caps(off) skip(on) color(green)
# type(text) intens(low) skip(on)
@ TYPE( INPUT) INTENS(LOW ) CAPS(On)  PADC('_') Color(green)
] TYPE( INPUT) INTENS(LOW ) CAPS(Off) PADC('_') color(green)
)body lmsg(msgl) smsg(msgs) window(65,11)
+                  Release information
+ %msgl%msgs++
+Release name:@projnm  +
+SD release  :@Z       +
+Status      :%Z+
+Userid      : &userid
+Description:
+ ]projtx1                                                    #
```

```
# ]projtx2                                                              #
# ]projtx3                                                              #
# ]projtx4                                                              #
)init
  .HELP = CHGHPRJU
  .zvars = '(sdrel status)'
)reinit
)proc
    ver(&projnm,nb)
    ver(&status,nb,list,T,R,A,P)
    ver(&projtx1,nb)
)end
```

## CHGPPROJ

```
)ATTR
/*************************************************************************/
/*                                                                       */
/*************************************************************************/
   # TYPE( INPUT) INTENS(LOW)  CAPS(ON ) JUST(RIGHT) PADC('_')
   ] TYPE( INPUT) INTENS(HIGH) CAPS(ON ) JUST(LEFT ) PAD('.')
   @ TYPE( INPUT) INTENS(LOW ) CAPS(ON)  JUST(LEFT ) PADC('_')
   $ TYPE(OUTPUT) INTENS(LOW ) CAPS(Off) JUST(LEFT ) PADC('_')
COLOR(GREEN)
   + TYPE(  TEXT) INTENS(LOW ) SKIP(ON)
)BODY  EXPAND(~~)
%-~-~- DDL-releases -~-~-
%COMMAND ===>_PCMD                                  ~ ~%SCROLL ===>_AMT +
% Datum
%   Chang-id   Date       Description
%   -------- - ---------- ----------------------------------------------
--------
)MODEL ROWS(SCAN)
]Z $Z       $Z$Z         $Z
)INIT
 .ZVARS='(LCMD projnm status credt projtx)'
  &LCMD=&Z
  &PCMD=''
  &AMT = CSR
  .HELP= CHGHPROJ
)REINIT
  IF (.MSG = ' ')
    &LCMD=' '
    REFRESH(LCMD)
)PROC
    VER(&LCMD,LIST,CH,D,GD,HI,I,M,NW,PR,S,ST,U)
)END
```

## CHGRCHGM

```
/* Rexx ************************************************************ */
/* Name     : CHGRNEWM                                               */
/* Purpose  : Connect a existing production member to a release       */
/* Parameter: parmsrch  - search argument                             */
/*            Parmtype  - objecttype                                 */
/*            Parmcrea  - creator/database/collection                */
/* ***************************************************************** */
Trace o
arg projnm status
Call CHGRVARS
address Ispexec "vGET (chgvddl)"
address Ispexec "vGET (chgvstat)"
address Ispexec "vGET (chgvdb2s)"
address Ispexec "vGET (chgvhlq)"
address Ispexec "vGET (chgvdb2l)"
Call init
If status <> 'T' then do
   erc=8
   emsg='Connect existing member not possible in status 'status
   Address Ispexec "vput (emsg)"
   return erc
End
Start_panel:
count=Ø
cmd  ='ANY'
msg  =''
prj1 =chgvhlq
lib1 ='DDL'
lib2 =''
lib3 =''
typ1 ='PROD'
address ISPEXEC
 "DISPLAY PANEL(ISREDMØ1)"
 erc=rc
 if erc>Ø then return erc
 "CONTROL ERRORS RETURN"
 "LMINIT DATAID(IN) PROJECT("prj1") ,
         GROUP1("lib1") GROUP2("lib2") GROUP2("lib3"),
         TYPE(PROD) ENQ(SHR)"
 if rc>Ø
 then do
    "SETMSG MSG("zerrmsg")"
    signal start_panel
 end
 "LMOPEN DATAID("in") OPTION(INPUT) LRECL(LR) RECFM(RM) ORG(O)"
 csr='ZCMD'; tmbr=''
/* ***************************************************************** *
 * select member
```

```
 * ********************************************************************* */
 select_mbr:
panel='ISRUDSM'
 "LMMDISP DATAID("in")  MEMBER("mem") STATS(YES) CURSOR("csr")
          PANEL("panel") TOP("tmbr") COMMANDS("cmd") FIELD(1)"
 erc = rc
 select
   when (zllib='1')  then choice=lib1
   when (zllib='2')  then choice=lib2
   when (zllib='3')  then choice=lib3
   when (zllib='4')  then choice=lib4
  otherwise do
   if erc=4 then do
      zmsg000s=
      zmsg000l='Dataset empty or wrong pattern'
      End
   if erc=8 then do
      zmsg000s=
      zmsg000l='Function ended by user'
      End
   if erc>8 then do
      zmsg000s='Error: 'erc
      zmsg000l='SEVERE ERROR occurred'
      End
   "SETMSG MSG(ISPZ000)"
   "LMCLOSE DATAID("in")"
   "LMFREE  DATAID("in")"
   "LMFREE  DATAID("out")"
   signal start_panel
   end
 end
 if choice='DDL'   then type='D'
 if choice='JCL'   then type='J'
 if choice='PROCS' then type='P'
 If choice='SYSIN' then type='S'
 mbrnm = strip(zlmember)
 Call Connect_controle /* member already connected? */
 erc=result
 If erc=111 then do
    udata='*Stop'
    zmsg000s='Connect error'
    zmsg000l='Member 'mbrnm' connected to release 'projnm
    "SETMSG MSG(ISPZ000)"
    "LMMDISP DATAID("in") OPTION(PUT) MEMBER("mbrnm") ZLUDATA("udata")"
    end
    else erc=0
 If erc=0 then do   /* Member in TEST-lib? */
   temp=chgvhlq'.'choice'.TEST'
   If sysdsn("'"temp'('mbrnm')'"'") = 'OK'
      Then erc=1
```

```
      Else do  /* Member in ACPT-lib? */
        temp=chgvhlq'.'choice'.ACPT'
        If sysdsn("'"temp'('mbrnm')'"'") = 'OK' then erc=1
        End
   If erc=1 then do
      csr='ZLLCMD'
      tmbr=mbrnm
      udata='*Stop'
      zmsg000s='Connect error'
      zmsg000l='Member 'mbrnm' connected, in libr. 'temp
      "SETMSG MSG(ISPZ000)"
      "LMMDISP DATAID("in") OPTION(PUT)
               MEMBER("mbrnm") ZLUDATA("udata")"
      msg=''
      signal select_mbr
      End
   if erc=0  then Call copy_member
   if erc<>0 then do
      udata='*Stop'
      zmsg000s='Copy error'
      zmsg000l='rc:'erc' from copy member 'mbrnm
      "SETMSG MSG(ISPZ000)"
      "LMMDISP DATAID("in") OPTION(PUT)
                   MEMBER("mbrnm") ZLUDATA("udata")"
      End
   /* Add member to picklist */
   If erc=0  then Call edit_picklist
   If erc<>0 then do
      udata = '*Stop'
      zmsg000s = 'Connect error'
      zmsg000l = 'Rc 'erc' from connect to release ' projnm
      "SETMSG MSG(ISPZ000)"
      "LMMDISP DATAID("in") OPTION(PUT)
                   MEMBER("mbrnm") ZLUDATA("udata")"
      End
   If (erc=0 & type='D') then do
      Call ddl_parms
      If erc>4 then do
         udata = '*Stop'
         zmsg000s = 'DDL-parm error'
         zmsg000l = 'Error changing DDL parms (DDL-move ok)'
         "SETMSG MSG(ISPZ000)"
         "LMMDISP DATAID("in") OPTION(PUT)
                   MEMBER("mbrnm") ZLUDATA("udata")"
         End
      Else erc=0
      End
End
If erc=0 then do
   Call CHGRLOG 'Release 'projnm' prod-member 'mbrnm' connected'
```

```
   zmsg000s = mbrnm' connected'
   zmsg000l = 'Member 'mbrnm' connected to release ' projnm
   udata = '*Ok'
   "SETMSG MSG(ISPZ000)"
   "LMMDISP DATAID("in") OPTION(PUT)
                    MEMBER("mbrnm") ZLUDATA("udata")"
End
Delstack
Signal select_mbr /* signal member selection */
/* ************************************************************ */
Init:
  X=outtrap(line.);
  Address TSO 'LISTUSER'
  X=outtrap('OFF');
  Do i=1 to line.0;
  x=pos('GROUP=',line.i,1)
  If x>0 then do
     racfgrp=substr(line.i,x+6,8)
     leave
     End;
  End;
  Select
    when racfgrp='?' then nop
    otherwise do
       emsg='RACF-group 'racfgrp' not authorised'
       erc=117
       "Ispexec vput (emsg)"
       exit erc
       End
    End
 Call CHGRCONN
 If result<>0 then do
    erc=result
    emsg='Error 'result' connecting to DB2'
    esigl=sigl
    Call SQL_error
 End
Return
Connect_controle:
  SQL_stmt="Select a.projnm From DB2CHGE.CHV002_connect a,
   DB2CHGE.CHV001_project b Where a.projnm=b.projnm
   and a.mbrnm='"mbrnm"' and a.mbrtyp='"type"' and b.status in
('T','A')"
  ADDRESS DSNREXX "EXECSQL DECLARE C1 CURSOR FOR S1"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc<>0 then Call SQL_error
  ADDRESS DSNREXX "EXECSQL PREPARE S1 FROM :SQL_stmt"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc<>0 then Call SQL_error
  ADDRESS DSNREXX "EXECSQL OPEN C1"
```

```
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  Do until erc<>0
     ADDRESS DSNREXX "EXECSQL FETCH C1 INTO :projnm"
     erc=sqlcode
     Select
        when erc=0 then do  /* member already connected */
          erc=111
          End
        when erc=100 then nop /* member not yet connected */
        when erc=-551 then do
          zmsg000l='User 'userid()' not authorised'
          zmsg000s='NOT AUTHORISED'
          "ISPEXEC SETMSG MSG(ISPZ000)"
          End
        otherwise Call SQL_error
     End
  End
  ADDRESS DSNREXX "EXECSQL CLOSE C1 "
  If sqlcode<>0 then Call SQL_error
  ADDRESS DSNREXX "EXECSQL COMMIT"
  If sqlcode<>0 then Call SQL_error
Return erc
Copy_member:
  "LMINIT DATAID(OUT) PROJECT("prj1") ,
          GROUP1("choice") TYPE(TEST) ENQ(SHR)"
  "LMCOPY FROMID("in")    FROMMEM("mbrnm"),
          TODATAID("out") TOMEM("mbrnm")"
  erc = rc
Return erc
DDL_parms:
  chgvtarg='P'
  "VPUT (chgvtarg) PROFILE"
  "EDIT DATAID("out") MEMBER("mbrnm") MACRO(CHGMDVAR)"
  erc=rc
Return erc
Edit_picklist:
  exein='N'
  If type='D' Then do
     exein='Y'
     mbrnm=strip(mbrnm)
     ddlseq=9
     If substr(mbrnm,3,1)='T' then ddlseq=4
     If substr(mbrnm,3,1)='V' then ddlseq=6
     If substr(mbrnm,length(mbrnm)-2,2)='FK' then ddlseq=9
     If substr(mbrnm,3,1)='V' then exein='N'
     End
     Else ddlseq=0
  SQL_stmt= "Insert into DB2CHGE.CHV002_CONNECT Select a.projnm
,'"mbrnm"'
       ,'"type"' ,'C' ,'"userid()"' ,current timestamp,'"exein"',"ddlseq"
```

```
      From DB2CHGE.CHV001_project a Where a.projnm='"projnm"' and
      a.updts= (Select max(b.updts) from DB2CHGE.CHV001_project B Where
      a.projnm=b.projnm)"
  ADDRESS DSNREXX "EXECSQL "SQL_stmt
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  Select
     when erc=0 then do
        ADDRESS DSNREXX "EXECSQL COMMIT"
        End
     when erc=-551 then do
          zmsg000l='User 'userid()' not authorised'
          zmsg000s='NOT AUTHORISED'
          "ISPEXEC SETMSG MSG(ISPZ000)"
          End
     when erc=-803 then do
        zmsg000l=mbrnm' connected'
        zmsg000s='Error 'erc
        "ISPEXEC SETMSG MSG(ISPZ000)"
        End
     Otherwise Call SQL_error
  End
Return erc
SQL_error:
   Say '*****************************************************'
  Say '* SQL-ERROR     'date(e)'         Progr:  CHGRCHGM  *'
   Say '*****************************************************'
  Say '*  Line-number REXX-Routine  =' esigl
  Say '*  SQLCODE     =' erc
  Say '*  SQLERRMC    =' emsg
  Say '*  SQL stmt    =' SQL_stmt
  ADDRESS DSNREXX "DISCONNECT"
  zmsg000l=emsg
  zmsg000s='SQL_error 'erc' on line 'esigl
  Address Ispexec "SETMSG MSG(ISPZ000)"
  Address Ispexec "vput (emsg)"
  Exit erc
Return
```

## CHGRCONN

```
/* Rexx ************************************************************ */
/* Name     : CHGRCONN                                               */
/* Purpose  : Connect to DB2                                         */
/* ***************************************************************** */
trace o
  address ispexec "vget (chgvdb2l)"
  address ispexec "vget (chgvdb2s)"
  address ispexec
  "LIBDEF ISPLLIB DATASET ID('"chgvdb2l"')"
```

```
  address tso
  'SUBCOM DSNREXX'  /* Is host command env avbl ? */
  erc=rc
  If erc then        /* If not then add it         */
     S_rc = RXSUBCOM('ADD','DSNREXX','DSNREXX')
     ADDRESS DSNREXX
     ADDRESS DSNREXX "CONNECT" chgvdb2s
     erc=sqlcode
  If erc=Ø then
     ADDRESS DSNREXX "EXECSQL SET CURRENT PACKAGESET='DSNREXUR'"
Return erc;
```

## CHGRLOG

```
/* Rexx ************************************************************ */
/* Name     : CHGRLOG                                                */
/* Purpose  : Log important processing DDL changes                   */
/* Log-file : chgvlog (ispf variable)                                */
/******************************************************************** */
trace o
parse arg txt
SIGNAL ON ERROR
mess = 'CHG ERROR Ø'
x = outtrap(var.)
y = msg('OFF')
mess = 'CHG ERROR 1 ALLOC'
Address Ispexec "vget (chgvlog)"
"ALLOC F(WBCHG) DA('"chgvlog"') MOD REUSE"
do while txt ¬= ''
  queue date(e) time() userid() left(txt,52)
  txt=substr(txt,53)
end
mess = 'CHG ERROR 2 IO'
'execio' queued() 'diskw WBCHG (finis)'
mess = 'CHG ERROR 3 FREE'
"FREE F(WBCHG)"
if queued() > Ø then pull x
x = outtrap('off')
exit Ø
ERROR:
  mess = mess||' RC='||RC
  y = outtrap(vvv.)
  "TSO send '"mess"' user(userid())"
  if vvv.Ø > Ø then,
    "TSO send '"date()||' '||time()||' '||mess"' user(userid()) save"
  "FREE F(WBCHG)"
return
```

## CHGRMEMB

```
/* Rexx ************************************************************ */
/* Name     : CHGRMEMB                                               */
/* Purpose  : Show/edit release member information                    */
/* Parameter: parmsrch  - search argument                             */
/*            Parmtype  - objecttype                                  */
/*            Parmcrea  - creator/database/collection                 */
/* Errors   : 108: DDL exein Y/N not for JCL-SYSIN                    */
/*            109: DDL seqnr not allowed for JCL-SYSIN                */
/*            110: error delete member                                */
/*            111: error connect/update projnm/exein                  */
/* ****************************************************************** */
trace o
address Ispexec
/*signal on error*/
Call init      /*  initialization    */
Call CHGRCONN
   If result<>0 Then do
      erc=result
      emsg='Error 'result' connecting to DB2'
      esigl=sigl
      Call SQL_error
   End
If mbrnm¬='' Then i=loc_data(mbrnm)
             else call crea_tbl       /* create ISPF-table   */
/*  show project-panel  */
"TBTOP CHGTMEMH"
Do while disp_rc<8
  "TBVCLEAR CHGTMEMH"
  If word(pcmd,1)='H'  Then i=loc_data('%') /* refresh */
  If word(pcmd,1)='L'  then do              /* locate  */
     "TBTOP    CHGTMEMH"
     mbrnm=word(pcmd,2)
     End
  Else  mbrnm='        '
  upper mbrnm
  "TBSARG  CHGTMEMH next namecond(mbrnm,ge)"
  "TBDISPL CHGTMEMH panel(CHGPMEMB)"
  disp_rc=rc
  If disp_rc>4 Then leave
  If ztdsels>0 Then call Proces lcmd
  "TBQUERY  CHGTMEMH POSITION(QCSR)"
End
exit erc
/* subroutines etc.------------------------------------------- */
Init:
   address Ispexec "vGET (projnm projtx)"
   address Ispexec "vGET (chgvddl)"
   address Ispexec "vGET (chgvstat)"
```

```
   address Ispexec "vGET (chgvdb2s)"
   address Ispexec "vGET (chgvhlq)"
   address Ispexec "vGET (chgvdb2l)"
   sqlcode=0
   lcmd=' '
   tbcr=''
   disp_rc=0
   pending=0
   current=0
   sysdt=substr(DATE(s),1,4) || '-' || ,
         substr(DATE(s),5,2) || '-' || ,
         substr(DATE(s),7,2)
   systm=substr(TIME(),1,2) || '.' || ,
         substr(TIME(),4,2) || '.' || ,
         substr(TIME(),7,2)
Return;
Proces:
   parse arg lcmd
   row=qcsr
   rcmd=' '
   pds=chgvhlq'.'mbrtype'.'statxt
   libr=chgvhlq'.'mbrtype'.'statxt'('mbrnm')'
   If tbcr='' Then Call Get_creator
   zmsg000l=''
   zmsg000s=''
   select
     when LCMD='B' Then do /* browse member */
          If SYSDSN(''''libr'''') <>'OK'
          Then do
             zmsg000l='Member 'mbrnm' not found in ' || libr
             zmsg000s='NOT FOUND'
             "ISPEXEC SETMSG MSG(ISPZ000)"
             End
          else "browse dataset('"libr"')"
          End
     when LCMD='D' Then do /* disconnect/delete member */
          question='Disconnect member 'mbrnm' from release 'projnm'?'
          call ask_sure question
          If answer='Y' Then do
             call sel_row
             call del_row
             question='Delete member 'mbrnm' from 'libr'?'
             call ask_sure question
             If answer='Y' Then call del_mbr
             End
          End
     when LCMD='E' Then do /* edit member */
          If SYSDSN(''''libr'''') <> 'OK'
          Then do
             zmsg000l=projnm' not found in ' || libr
```

```
            zmsg000s='Fout 100'
            "ISPEXEC SETMSG MSG(ISPZ000)"
            End
         else "edit dataset('"libr"')"
         End
      when LCMD='S' Then do /* browse member */
         If SYSDSN(''''libr'''') <> 'OK'
         Then do
            zmsg000l=projnm' not found in ' || libr
            zmsg000s='Fout 100'
            "ISPEXEC SETMSG MSG(ISPZ000)"
            End
         else "edit dataset('"libr"')"
         End
      when LCMD='U' Then do
         oprojnm=projnm
         Call sel_row
         "addpop row(1) column(1)"
         "display panel(CHGPMEMU)"
         "rempop"
         If mbrtyp='S' & ddlseq>0 Then do
            zmsg000l='DDL sequence not allowed for SYSIN'
            zmsg000s='Fout 109'
            "ISPEXEC SETMSG MSG(ISPZ000)"
            Return
            End
         Else do
        question='Change info of member 'mbrnm' from release 'projnm'?'
            call ask_sure question
            If answer='Y' Then do
               Call upd_row
               If erc=0 Then i=loc_data('%')
               "TBTOP CHGTMEMH"
               End
         End
      End
      otherwise Return ;
    End
Return;
sel_row:
  SQL_stmt="Select mbrnm,mbrtyp,srccd,userid,date(updts),time(updts),
        exein,ddlseq From DB2CHGE.CHV002_connect A Where
projnm='"projnm"'
        and mbrnm='"mbrnm"'"
  ADDRESS DSNREXX "EXECSQL DECLARE C3 CURSOR FOR S3"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> 0 Then  Call SQL_error
  ADDRESS DSNREXX "EXECSQL PREPARE S3 FROM :SQL_stmt"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> 0 Then  Call SQL_error
```

```
  ADDRESS DSNREXX "EXECSQL OPEN C3"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> Ø Then  Call SQL_error
  ADDRESS DSNREXX "EXECSQL FETCH C3 INTO
     :mbrnm, :mbrtyp, :srccd, :userid, :upddt, :updtd, :exein, :ddlseq"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  Select
     when erc=Ø Then do  /* row found */
          mbrnm =strip(mbrnm )
          mbrtyp=strip(mbrtyp)
          srccd=strip(srccd)
          userid =strip(userid )
          upddt =strip(upddt )
          updtd =strip(updtd )
          exein =strip(exein )
          ddlseq=Translate(ddlseq,'','.')
       End
     when erc=1ØØ Then do /* NOT FOUND       */
       zmsgØØØl=emsg
       zmsgØØØs='NOT FOUND'
       "ISPEXEC SETMSG MSG(ISPZØØØ)"
       Return erc
       End
     when erc=-551 then do
       zmsgØØØl='User 'userid()' not authorised'
       zmsgØØØs='NOT AUTHORISED'
       "ISPEXEC SETMSG MSG(ISPZØØØ)"
       End
     otherwise do
       zmsgØØØl=emsg
       zmsgØØØs=''
       "ISPEXEC SETMSG MSG(ISPZØØØ)"
       Return erc
     End
  End
  ADDRESS DSNREXX "EXECSQL CLOSE C3"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> Ø Then  Call SQL_error

  ADDRESS DSNREXX "EXECSQL COMMIT"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> Ø Then  Call SQL_error
Return Ø
upd_row:
  SQL_stmt= "Update DB2CHGE.CHVØØ2_connect set projnm ='"projnm"'
     ,exein='"exein"',ddlseq="ddlseq" Where projnm='"oprojnm"' and
     mbrnm='"mbrnm"' and mbrtyp='"mbrtyp"'"
  ADDRESS DSNREXX "EXECSQL "SQL_stmt
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  ADDRESS DSNREXX "EXECSQL COMMIT"
```

```
  Select
     when erc=Ø Then do
        If oprognm<>prognm Then "tbdelete CHGTMEMH"
                           else "tbput CHGTMEMH order"
        zmsgØØØl=mbrnm || ' CHANGED'
        zmsgØØØs=mbrnm || ' CHANGED'
        "ISPEXEC SETMSG MSG(ISPZØØØ)"
        projnm=oprojnm
        End
     when erc=1ØØ Then do
        zmsgØØØl='Member 'mbrnm' NOT FOUND'
        zmsgØØØs='NOT FOUND'
        "ISPEXEC SETMSG MSG(ISPZØØØ)"
        End
     when erc=-551 then do
       zmsgØØØl='User 'userid()' not authorised'
       zmsgØØØs='NOT AUTHORISED'
       "ISPEXEC SETMSG MSG(ISPZØØØ)"
       End
     Otherwise do
        zmsgØØØl=erc || emsg
        zmsgØØØs='Error 111'
        "ISPEXEC SETMSG MSG(ISPZØØØ)"
        Return erc
        End
  End
Return Ø
Del_row:  /* disconnect member */
  SQL_stmt= "Delete from DB2CHGE.CHVØØ2_connect "
  SQL_stmt=SQL_stmt ||,
  " WHERE projnm='"projnm"' and mbrnm='"mbrnm"' and mbrtyp='"mbrtyp"'"
  ADDRESS DSNREXX "EXECSQL "SQL_stmt
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  Select
     when erc=Ø Then do
        "tbdelete CHGTMEMH"
        Call CHGRLOG 'Member 'mbrnm' disconnected from release 'projnm
        zmsgØØØs=mbrnm||' DISCONNECTED'
        zmsgØØØl=mbrnm || ' disconnected from release 'projnm
        "ISPEXEC SETMSG MSG(ISPZØØØ)"
        End
     when erc=1ØØ Then do
        zmsgØØØl=erc || emsg
        zmsgØØØs=mbrnm || ' NOT FOUND'
        "ISPEXEC SETMSG MSG(ISPZØØØ)"
        End
     when erc=-551 then do
       zmsgØØØl='User 'userid()' not authorised'
       zmsgØØØs='NOT AUTHORISED'
       "ISPEXEC SETMSG MSG(ISPZØØØ)"
```

```
       End
     Otherwise Call SQL_error
  End
Return Ø
Del_mbr:  /* delete member */
 "CONTROL ERRORS Return"
 "LMINIT DATAID(IN) DATASET('"pds"') ENQ(EXCLU)"
 If rc=Ø Then do
   "LMOPEN DATAID("in") OPTION(OUTPUT)"
   "LMMDEL dataid("in") MEMBER("mbrnm")"
   erc=rc
   "LMFREE DATAID("in")"
   Select
     When erc=Ø Then
        Call CHGRLOG 'Member 'mbrnm' deleted from 'pds
     When erc=8 Then do
          zmsgØØØs=mbrnm||' NOTFOUND'
          zmsgØØØl=mbrnm || ' not found in libr. 'pds
          "ISPEXEC SETMSG MSG(ISPZØØØ)"
          End
     Otherwise exit(erc)
   End
 End
Return Ø
ask_sure:
  parse arg question
  "addpop poploc(mbrnm)"
  "display panel(CHGPMTOK)"
  i=answer='Y'
  "rempop"
Return i
loc_data:
  Call Get_release_info
   SQL_stmt="Select mbrnm,mbrtyp,srccd,userid,date(updts),time(updts),
         exein,ddlseq From DB2CHGE.CHVØØ2_CONNECT A Where
projnm='"projnm"'
         Order by mbrtyp asc, ddlseq asc, mbrnm asc"
  SQL_stmt = Translate(SQL_stmt,'  ',' ')
  ADDRESS DSNREXX "EXECSQL DECLARE C2 CURSOR FOR S2"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> Ø Then  Call SQL_error
  ADDRESS DSNREXX "EXECSQL PREPARE S2 FROM :SQL_stmt"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> Ø Then  Call SQL_error
  ADDRESS DSNREXX "EXECSQL OPEN C2"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> Ø Then  Call SQL_error
  "tbcreate CHGTMEMH
   names(rcmd,mbrnm,mbrtype,srccd,exein,userid,upddt,updtd,mbrstat,ddlseq
        ,srctxt) replace nowrite"
```

```
"tbsort CHGTMEMH fields(mbrnm,C,A,mbrtype,C,A,ddlseq,N,A)"
t=Ø
Do until erc<>Ø
   ADDRESS DSNREXX "EXECSQL FETCH C2 INTO
    :mbrnm, :mbrtyp, :srccd, :userid, :upddt, :updtd, :exein, :ddlseq"
    erc=sqlcode;esigl=sigl;emsg=sqlerrmc
   Select
      when erc=Ø Then do  /* row found  */
        t=t+1
        mbrnm=strip(mbrnm)
        ddlseq=Translate(ddlseq,'','.')
        Select
          When mbrtyp='D' Then mbrtype='DDL'
          When mbrtyp='J' Then mbrtype='JCL'
          When mbrtyp='P' Then mbrtype='PROCS'
          When mbrtyp='S' Then mbrtype='SYSIN'
          Otherwise nop
        End
        Select
          When strip(srccd)='N' then srctxt='New'
          When strip(srccd)='C' then srctxt='Chg'
          Otherwise srctxt='?'
        End
        libr = chgvhlq'.'mbrtype'.'statxt'('mbrnm')'
        mbrstat='Ok'
        If SYSDSN(''''libr'''') <> 'OK' Then mbrstat='Error'
        "tbadd CHGTMEMH order"
        mbrnm.t  =mbrnm
        mbrtype.t=mbrtype
        srccd.t =srccd
        srctxt.t =srctxt
        userid.t =userid
        upddt.t  =upddt
        updtd.t  =updtd
        exein.t  =exein
        ddlseq.t =ddlseq
        End
      when erc=1ØØ & t>Ø Then NOP
      when erc=1ØØ & t=Ø Then do
        /* NOT FOUND              */
        msg='No members connected to release 'projnm
        Address Ispexec "vput (erc msg)"
        End
   when erc=-551 then do
        zmsgØØØl='User 'userid()' not authorised'
        zmsgØØØs='NOT AUTHORISED'
        "ISPEXEC SETMSG MSG(ISPZØØØ)"
        End
   when erc<>Ø  Then do
        Address Ispexec "vput (erc msg)"
```

```
          End
        Otherwise nop
      End
  End
  ADDRESS DSNREXX "EXECSQL CLOSE C2"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> Ø Then  Call SQL_error
  ADDRESS DSNREXX "EXECSQL COMMIT"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> Ø Then  Call SQL_error
Return Ø
Get_release_info:
  SQL_stmt= ,
    "Select projnm ,date(updts), time(updts), userid " ||,
    ",status, SDREL, projtx From  DB2CHGE.CHVØØ1_project A " ||,
    " Where UCASE(A.projnm) = '"projnm"' and updts = " ||,
    " (Select max(updts) From DB2CHGE.CHVØØ1_PROJECT B " || ,
    " Where A.projnm=B.projnm) Order by A.projnm"
  ADDRESS DSNREXX "EXECSQL DECLARE C1 CURSOR FOR S1"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> Ø Then  Call SQL_error
  ADDRESS DSNREXX "EXECSQL PREPARE S1 FROM :SQL_stmt"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> Ø Then  Call SQL_error
  ADDRESS DSNREXX "EXECSQL OPEN C1"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> Ø Then  Call SQL_error
  ADDRESS DSNREXX "EXECSQL FETCH C1 INTO
           :projnm, :credt, :cretd, :userid, :status, :sdrel, :projtx"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  Select
     when erc=Ø Then do  /* row found    */
       If status='T' Then statxt='TEST'
       If status='A' Then statxt='ACPT'
       If status='P' Then statxt='PROD'
       End
     when erc=1ØØ Then do /* NOT FOUND       */
       zmsgØØØl=erc || emsg
       zmsgØØØs='NOT FOUND'
       "ISPEXEC SETMSG MSG(ISPZØØØ)"
       Return erc
       End
     when erc=-551 then do
       zmsgØØØl='User 'userid()' not authorised'
       zmsgØØØs='NOT AUTHORISED'
       "ISPEXEC SETMSG MSG(ISPZØØØ)"
       End
     otherwise do
       zmsgØØØl=erc || emsg
       zmsgØØØs=emsg
```

```
        "ISPEXEC SETMSG MSG(ISPZ000)"
        Return erc
      End
  End
  ADDRESS DSNREXX "EXECSQL CLOSE C1 "
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> 0 Then  Call SQL_error
  ADDRESS DSNREXX "EXECSQL COMMIT"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> 0 Then  Call SQL_error
Return erc
Get_creator:
  SQL_stmt= ,
    "Select creator From SYSIBM.SYSTABLES Where name='"STRIP(mbrnm)"'"
  ADDRESS DSNREXX "EXECSQL PREPARE S1 FROM :SQL_stmt"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> 0 Then  Call SQL_error
  ADDRESS DSNREXX "EXECSQL OPEN C1"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> 0 Then  Call SQL_error
  ADDRESS DSNREXX "EXECSQL FETCH C1 INTO :tbcr"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  Select
     when erc=0 Then do  /* ROW FOUND  */
       End
     when erc=100 Then do /* NOT FOUND       */
       ADDRESS DSNREXX "EXECSQL CLOSE C1 "
       Return erc
       End
     when erc=-551 then do
       zmsg000l='User 'userid()' not authorised'
       zmsg000s='NOT AUTHORISED'
       "ISPEXEC SETMSG MSG(ISPZ000)"
       End
     otherwise nop
  End
  ADDRESS DSNREXX "EXECSQL CLOSE C1 "
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> 0 Then  Call SQL_error
  ADDRESS DSNREXX "EXECSQL COMMIT"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> 0 Then  Call SQL_error
Return erc
SQL_error:
  Say '*****************************************************'
  Say '* SQL-ERROR     'date(e)'         Progr:  CHGRMEMB   *'
  Say '*****************************************************'
  Say '*  Line-number REXX-Routine  =' esigl
  Say '*  SQLCODE     =' erc
  Say '*  SQLERRMC    =' emsg
```

```
  Say '*  SQL stmt    =' SQL_stmt
  ADDRESS DSNREXX "DISCONNECT"
  zmsg000l=emsg
  zmsg000s='SQL_error 'erc' on line 'esigl
  Address Ispexec "SETMSG MSG(ISPZ000)"
  Address Ispexec "vput (emsg)"
  Exit erc
Return
```

## CHGRNEWM

```
/* Rexx ************************************************************* */
/* Name     : CHGRNEWM                                                */
/* Purpose  : Connect a new member to a release                       */
/* Errors   : 111: member already connected to a release              */
/*            112: new members already exists in production           */
/* Parameter: parmsrch  - search argument                             */
/*            Parmtype  - objecttype                                  */
/*            Parmcrea  - creator/database/collection                 */
/* ****************************************************************** */
Trace o
arg projnm status
Call CHGRVARS
address Ispexec "vGET (chgvddl)"
address Ispexec "vGET (chgvstat)"
address Ispexec "vGET (chgvdb2s)"
address Ispexec "vGET (chgvhlq)"
address Ispexec "vGET (chgvdb2l)"
Call init
If status='P' then do
   erc=8
   emsg='Cannot connect in status 'status
   "Ispexec vput (emsg)"
   return erc
   End
Delstack
Dropbuf
Start_panel:
count=0
cmd ='ANY'
msg =''
prj1=chgvhlq
lib1='DDL'
lib2=''
lib3=''
typ1='TEST'
zmsg000s=''
zmsg000l=''
address ISPEXEC
```

```
 "DISPLAY PANEL(ISREDM01)"
 if rc > 0 then return
 "CONTROL ERRORS RETURN"
 "LMINIT DATAID(IN) PROJECT("prj1") ,
         GROUP1("lib1") GROUP2("lib2") GROUP2("lib3"),
         TYPE("typ1") ENQ(SHR)"
 if rc > 0
 then do
    "SETMSG MSG("zerrmsg")"
    signal start_panel
 end
 "LMOPEN DATAID("in") OPTION(INPUT) LRECL(LR) RECFM(RM) ORG(O)"
 csr='ZCMD'; tmbr=''
 select_mbr:
panel='ISRUDSM'
 "LMMDISP DATAID("in")  MEMBER("mem") STATS(YES) CURSOR("csr")
          PANEL("panel") TOP("tmbr") COMMANDS("cmd") FIELD(1)"
 If rc>4 then do
    exit rc  /* stop connecting  */
    End
 Select
   when (zllib='1')    then choice=lib1
   when (zllib='2')    then choice=lib2
   when (zllib='3')    then choice=lib3
   when (zllib='4')    then choice=lib4
  otherwise do
   if erc=4 then do
      zmsg000s=
      zmsg000l='Dataset empty or wrong pattern'
      End
   if erc=8 then do
      zmsg000s=
      zmsg000l='Function ended by user'
      End
   if erc>8 then do
      zmsg000s='Error: 'erc
      zmsg000l='SEVERE ERROR occurred'
      End
   "SETMSG MSG(ISPZ000)"
   "LMCLOSE DATAID("in")"
   "LMFREE  DATAID("in")"
   "LMFREE  DATAID("out")"
   signal start_panel
   end
 end
 if choice='DDL'   then type='D'
 if choice='JCL'   then type='J'
 if choice='PROCS' then type='P'
 if choice='SYSIN' then type='S'
 mbrnm=strip(zlmember)
```

```
Call Connect_check
erc=result
if erc=111 then do
   udata='*Stop'
   zmsg000s='Connect error'
   zmsg000l='Member 'mbrnm ' connected to release 'projnm
   "SETMSG MSG(ISPZ000)"
   "LMMDISP DATAID("in") OPTION(PUT)
                  MEMBER("mbrnm") ZLUDATA("udata")"
   end
   else erc=0
If erc=0 then do
   Call Exists /* member exitst in acpt or prod */
   erc=result
   if erc=112 then do
      udata='*Stop'
      zmsg000s='Member exists'
      zmsg000l=mbrnm' is no NEW object: member exists in production'
      "SETMSG MSG(ISPZ000)"
      "LMMDISP DATAID("in") OPTION(PUT)
                  MEMBER("mbrnm") ZLUDATA("udata")"
      end
      else erc=0
   end
If erc=0 then do
  /* COPY member to TEST libr. */
  If typ1<>'TEST' & typ1<>'ACPT' & typ1<>'PROD' then do
     temp=chgvhlq'.'choice'.TEST'
     If sysdsn("'"temp'('mbrnm')'"'") <> 'OK' then do
        Call copy_member
        erc=result
        If erc<>0 then do
           udata='*Stop'
           zmsg000s='Copy error'
           zmsg000l=mbrnm ' found in TEST-libr. 'projnm
           "SETMSG MSG(ISPZ000)"
           "LMMDISP DATAID("in") OPTION(PUT)
                    MEMBER("mbrnm") ZLUDATA("udata")"
           End
        End
     End
  /* ADD member to picklist */
  if erc=0  then Call edit_picklist
  if erc<>0 then do
     zmsg000s='Connect error'
     udata='*Ok'
     zmsg000l='Rc:'erc' during connect 'mbrnm' to release 'projnm
     "SETMSG MSG(ISPZ000)"
     "LMMDISP DATAID("in") OPTION(PUT)
                  MEMBER("mbrnm") ZLUDATA("udata")"
```

```
   End
End
If erc=Ø then do
   Call CHGRLOG 'Release 'projnm' test-member 'mbrnm' connected'
   zmsgØØØs='Connected to 'projnm
   zmsgØØØl='Member 'mbrnm' connected to release ' projnm
   udata='*Ok'
   "SETMSG MSG(ISPZØØØ)"
   "LMMDISP DATAID("in") OPTION(PUT)
                    MEMBER("mbrnm") ZLUDATA("udata")"
End
Signal select_mbr /* select next member */
Return erc
/* ************************************************************* */
Init:
  X=outtrap(line.);
  Address TSO 'LISTUSER'
  X=outtrap('OFF');
  Do i=1 to line.Ø;
  x=pos('GROUP=',line.i,1)
  If x>Ø then do
     racfgrp=substr(line.i,x+6,8)
     leave
     End;
  End;
  Select
    when racfgrp='?' then nop
    otherwise do
       emsg='RACF-group 'racfgrp' not authorised'
       erc=117
       "Ispexec vput (emsg)"
       exit erc
       End
    End
  Call CHGRCONN
  If result<>Ø then do
      erc=result
      emsg='Error 'result' connecting to DB2'
      esigl=sigl
      Call SQL_error
  End
Return
Exists:
   temp=chgvhlq'.'choice'.PROD'
   if sysdsn("'"temp'('mbrnm')'"'")='OK'
      then erc=112 /* member exists */
      else erc=Ø
Return erc
Copy_member:
  "LMINIT DATAID(OUT) PROJECT("prj1") ,
```

```
          GROUP1("choice") TYPE(TEST) ENQ(SHR)"
  "LMCOPY FROMID("in")    FROMMEM("mbrnm"),
          TODATAID("out") TOMEM("mbrnm")"
  erc=rc
  "LMCLOSE DATAID("in")"
  "LMFREE  DATAID("in")"
  "LMCLOSE DATAID("out")"
  "LMFREE  DATAID("out")"
Return erc
Edit_picklist:
  If type='D' then ddlseq=8
              else ddlseq=Ø
  If type='D' then exein='Y'
              else exein='N'
  SQL_stmt= "Insert into DB2CHGE.CHVØØ2_connect Select
a.projnm,'"mbrnm"'
      ,'"type"' ,'N' ,'"userid()"' ,current timestamp,'"exein"'
,"ddlseq"
       From DB2CHGE.CHVØØ1_PROJECT a Where a.projnm='"projnm"'
       and a.updts= (Select max(b.updts) from DB2CHGE.CHVØØ1_PROJECT B
       Where a.projnm=b.projnm)"
  ADDRESS DSNREXX "EXECSQL "SQL_stmt
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  Select
     when erc=Ø then do
        ADDRESS DSNREXX "EXECSQL COMMIT"
        End
     when erc=-551 then do
          zmsgØØØl='User 'userid()' not authorised'
          zmsgØØØs='NOT AUTHORISED'
          "ISPEXEC SETMSG MSG(ISPZØØØ)"
          End
     when erc=-8Ø3 then do
          csr='ZLLCMD'
          tmbr=mbrnm
          zmsgØØØs='Connect error'
          zmsgØØØl='Member 'mbrnm' connected (See 'temp')'
         "SETMSG MSG(ISPZØØØ)"
          udata='*Stop'
          "LMMDISP DATAID("in") OPTION(PUT)
                     MEMBER("mbrnm") ZLUDATA("udata")"
          msg=' '
          /* New member in ACPT connected */
          If typ1='ACPT' then exit Ø
          signal select_mbr
        End
     Otherwise Call SQL_error
  End
Return erc
```

```
Connect_check: /* check if member is connected to release */
  SQL_stmt="Select p.projnm From DB2CHGE.CHV002_CONNECT k
  ,DB2CHGE.CHV001_PROJECT p Where p.projnm=k.projnm and p.status in
  ('T','A') and mbrnm='"mbrnm"' and mbrtyp='"type"'"
  ADDRESS DSNREXX "EXECSQL DECLARE C1 CURSOR FOR S1"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> 0 then Call SQL_error
  ADDRESS DSNREXX "EXECSQL PREPARE S1 FROM :SQL_stmt"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> 0 then Call SQL_error
  ADDRESS DSNREXX "EXECSQL OPEN C1"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  If erc <> 0 then Call SQL_error
  ADDRESS DSNREXX "EXECSQL FETCH C1 INTO :projnm"
  erc=sqlcode;esigl=sigl;emsg=sqlerrmc
  Select
     when erc=0 then do  /* row found      */
       erc=111 /* member already connected */
       End
     when erc=100 then nop  /* member not yet connected */
     when erc=-551 then do
          zmsg000l='User 'userid()' not authorised'
          zmsg000s='NOT AUTHORISED'
          "ISPEXEC SETMSG MSG(ISPZ000)"
          End
     otherwise Call SQL_error
  End
  ADDRESS DSNREXX "EXECSQL CLOSE C1 "
  If sqlcode <> 0 then Call SQL_error
  ADDRESS DSNREXX "EXECSQL COMMIT"
  If sqlcode <> 0 then Call SQL_error
Return erc
SQL_error: /* Ophalen DB2-errorvariabelen (SQLCA) igv errorconditions */
   Say '*****************************************************'
  Say '* SQL-ERROR     'date(e)'         Progr:  CHGRNEWM  *'
   Say '*****************************************************'
  Say '*  Line-number REXX-Routine  =' esigl
  Say '*  SQLCODE     =' erc
  Say '*  SQLERRMC    =' emsg
  Say '*  SQL stmt    =' SQL_stmt
  ADDRESS DSNREXX "DISCONNECT"
  zmsg000l=emsg
  zmsg000s='SQL_error 'erc' on line 'esigl
  Address Ispexec "SETMSG MSG(ISPZ000)"
  Address Ispexec "vput (emsg)"
  Exit erc
Return
```

## CHGROVER

```
/* Rexx ************************************************************ */
```

```
/* Name     : CHGROVER                                                */
/* Purpose  : Move DB2 release to next/previous stage                 */
/* Parameter: Parmsrch  - search argument                             */
/*            Parmtype  - objecttype                                  */
/*            Parmcrea  - creator/database/collection                 */
/* Errors   : 108: choosen stage not possible                         */
/*            109: members missing on library                         */
/*            110: error changing DDL-parms (CHGMDVAR)                */
/*            111: release contains no members in CHT002              */
/*            112: error while moving old version to FALLBACK lib     */
/*            113: error moving release members                       */
/*            114: error update release status CHT001                 */
/*            115: error update srccd in CHT002                       */
/*            116: error changing memberdata (CHGMCVAR)               */
/*            117: error check RACF-group                             */
/* ********************************************************************* */
Trace o
arg projnm
projnm=strip(projnm)
address Ispexec "vGET (chgvddl)"
address Ispexec "vGET (chgvstat)"
address Ispexec "vGET (chgvdb2s)"
address Ispexec "vGET (chgvhlq)"
address Ispexec "vGET (chgvdb2l)"
/* ************* S T A R T    M A I N L I N E ********************** */
Begin_loop:
Call Init
Call Get_release_info
If erc=0 then Call Dis_panel
If erc=0 then Call Check
If erc>0 then signal begin_loop
txt0='==> Start processing'
ADDRESS DSNREXX "EXECSQL COMMIT"
"ISPEXEC CONTROL DISPLAY LOCK"
"ISPEXEC DISPLAY PANEL(CHGPOVER)"
If erc=0 then Call Read_members  /* read connected members         */
If erc=0 then Call Check_members /* check existence of members     */
   /* ==> Start critical updates <== */
   If erc=0 then do /* save old members to FALLBACK lib */
                 Call Fallback_members
                 End
   If erc=0 then do /* move connected members */
                 Call Move_members
                 End
   If erc=0 then do /* release status wijzigen */
                 Call Upd_release_status
                 End
   If erc=0 then do /* change srccd per member  (new/change) */
                 Call Upd_member_bron
                 End
```

```
   If erc=Ø then do /* Update variables   */
                 Call Upd_vars
                 ADDRESS DSNREXX "EXECSQL COMMIT"
                 End
   If erc=Ø then do /* Delete old total-DDL member */
                 Call Del_DDLTOT_oud
                 End
   If erc=Ø then do /* Make total DDL-file */
                 Call Make_DDLTOT_member
                 End
If erc=Ø then do
   txt9='==> Promotion ok, rc='erc
    ADDRESS DSNREXX "EXECSQL COMMIT"
   "ISPEXEC CONTROL DISPLAY LOCK"
   "ISPEXEC DISPLAY PANEL(CHGPOVER)"
   Call CHGRLOG 'Release 'projnm' to status 'statto' ('vmbr' members)'
   End
   Else Call SQL_error
Address Ispexec "vput (erc emsg)"
Address Ispexec "LMFREE  DATAID("in")"
Address Ispexec "LMFREE  DATAID("out")"
Address Ispexec "LMFREE  DATAID("ddltot")"
Address Ispexec "LMCLOSE DATAID("ddltot")"
Address Ispexec "LMFREE  DATAID("racf")"
Signal begin_loop
return erc
/* ************* E N D    M A I N L I N E ************************ */
Init:
  /* check RACF-group */
  X=outtrap(line.);
  Address TSO 'LISTUSER'
  X=outtrap('OFF');
  Do i=1 to line.Ø;
  x=pos('GROUP=',line.i,1)
  If x>Ø then do
     racfgrp=substr(line.i,x+6,8)
     leave
     End;
  End;
  Select
    when racfgrp='?' then nop
    otherwise do
       emsg='RACF-group 'racfgrp' not authorised'
       erc=117
       Address Ispexec "vput (emsg)"
       exit erc
       End
    End
  Call CHGRCONN
  If result<>Ø then do
```

```
      erc=result
      emsg='Error 'result' connecting to DB2'
      esigl=sigl
      Call SQL_error
    End
    sysdt=substr(DATE(s),1,4) || '-' || ,
          substr(DATE(s),5,2) || '-' || ,
          substr(DATE(s),7,2)
    systm=substr(TIME(),1,2) || '.' || ,
          substr(TIME(),4,2) || '.' || ,
          substr(TIME(),7,2)
    systs=sysdt || '-' || systm
    nddl=0 /* total connected DDL members */
    nmbr=0 /* total processed members */
    vmbr=0 /* total moved members */
Return
/*********************************************************************/
Dis_panel:
    emsg=''
    "Ispexec display panel(CHGPOVER) cursor(chgvtarg)"
    if rc>4 then exit 0
    Address Ispexec "vget (chgvtarg) profile"
    emsg=''
    txt0=''
    txt9=''
    Select
      when chgvtarg='R' then do
           statto='T'     /* new status         */
           libto ='TEST'
           libfrom ='ACPT'
           end
      when chgvtarg='T' then do
           statto='T'     /* new status         */
           libto='TEST'
           libfrom ='PROD'
           end
```

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

*Loet Polkamp*
*Database Administrator (The Netherlands)*

## DB2 LUW – new table functions for the operating environment

Have you seen the four new User Defined Functions that came along in DB2 UDB V8.2.2 (V8.1 FP9)? They allow you to obtain system information from SQL statements where previously you had to issue system commands. I found these UDFs on the developer works Web site – a great place to browse for new stuff.

The four UDFs are SYSPROC.ENV_GET_INST_INFO for instance information, SYSPROC.ENV_GET_PROD_INFO for environment information, SYSPROC.ENV_GET_SYS_INFO for system information, and reg_list_variables for registry variables. Let's look at each of these in turn.

The SQL in this article was run on a Windows 2000 Professional system running DB2 UDB 8.2. FP10 for LUW.

If you are not connected to a database when you issue the commands shown, a connection will be made for you to the database that is defined in the DB2DBDFT registry variable. If a database is not defined and you are not connected, then you will receive an SQL1328N error message if you try to execute the SQL.

The first UDF is SYSPROC.ENV_GET_INST_INFO. This gives me instance information in nine columns. To list all the columns, use:

```
>db2 describe select * from table(SYSPROC.ENV_GET_INST_INFO()) as s
```

An example of the query is:

```
>db2 select substr(inst_name,1,10) as "inst_name", is_inst_partitionable
as "IIP", num_dbpartitions as "num_part", inst_ptr_size as "instptr",
substr(release_num,1,10) as "rel_number", substr(service_level,1,15) as
"service_level", substr(bld_level,1,10) as "bld_level", substr(ptf,1,10)
as "ptf", fixpack_num as "Fixpak_NUM" from
table(SYSPROC.ENV_GET_INST_INFO()) as s

INST_NAME  IIP    NUM_PART    INSTPTR     REL_NUMBER SERVICE_LEVEL
```

```
BLD_LEVEL  PTF        FIXPAK_NUM
---------- ------ ---------- ---------- ---------- ------------- ------
---- ---------- -----------
DB2             1           1          32 Ø3Ø4Ø1Ø6   DB2 v8.1.1Ø.812
sØ5Ø811    WR21362             1Ø
```

This gives me, amongst other things, the instance name, partition number, and build information. I could store all of these values. What this doesn't tell me is where DB2 is installed, which is available when I issue the **db2level** command.

The second UDF is SYSPROC.ENV_GET_PROD_INFO. This gives me environment information. An example of the query is:

```
>db2 select * from table (sysproc.env_get_prod_info()) as s

INSTALLED_PROD             IS_LICENSED PROD_RELEASE
-------------------------- ---------- -------------------------
ESE                                 1 8.2
```

I think I can get more information from the **db2level** and **db2licm** commands.

The third UDF is SYSPROC.ENV_GET_SYS_INFO. This gives me system information. Have you ever been asked to confirm the operating system that you are running DB2 on? Or how about if you need to know when the latest service pack was applied to your operating system? The UDF returns seven columns, as shown below. It's a long query to type in, but the substr clauses make the output easier to read.

An example of the query is:

```
>db2 select substr(os_name,1,1Ø) as "osname", substr(os_version,1,1Ø) as
"osversion", substr(os_release,1,2Ø) as "release",
substr(host_name,1,2Ø) as "hostname", total_cpus,
configured_cpus,total_memory from table(sysproc.env_get_sys_info()) as s

OSNAME     OSVERSION  RELEASE              HOSTNAME
TOTAL_CPUS  CONFIGURED_CPUS TOTAL_MEMORY
---------- ---------- -------------------- -------------------- --------
-- ------------- ------------
WIN32_NT   5.Ø        Service Pack 4       XXX
1               1         2Ø47
```

I can get operating system name, version, and service pack level, as well as the number of CPUs and the memory installed. I can store this information along with a timestamp to tell me when a new service pack was installed or when the memory was upgraded.

The fourth UDF is reg_list_variables. This gives me registry variable information. An example of the query is:

```
>db2 select dbpartitionnum as "dbnum", substr(reg_var_name,1,3Ø) as
"Reg_var_name", substr(reg_var_value,1,5Ø) as "Reg_var_value",
is_aggregate as "IA", substr(aggregate_name,1,2Ø) as "Aggregate_name",
level from table(reg_list_variables()) as s order by 1,2

DBNUM  REG_VAR_NAME                    REG_VAR_VALUE
IA     AGGREGATE_NAME       LEVEL
------ ------------------------------ --------------------------------
--------------- ------ ------------------- -----
     Ø DB2_DOCHOST                    localhost
Ø -                    I
     Ø DB2_DOCPORT                    51ØØØ
Ø -                    G
     Ø DB2_DXX_PATHS_ALLOWED_READ     ANY
Ø -                    I
     Ø DB2_DXX_PATHS_ALLOWED_WRITE    ANY
Ø -                    I
     Ø DB2_EXTSECURITY                NO
Ø -                    G
     Ø DB2_RR_TO_RS                   YES
Ø -                    I
     Ø DB2ACCOUNTNAME                 XXX\db2admin
Ø -                    I
     Ø DB2ADMINSERVER                 DB2DASØØ
Ø -                    G
     Ø DB2COMM                        TCPIP
Ø -                    I
     Ø DB2DBDFT                       sample
Ø -                    I
     Ø DB2INSTDEF                     DB2
Ø -                    G
     Ø DB2INSTOWNER                   XXX
Ø -                    I
     Ø DB2INSTPROF                    C:\PROGRA~1\IBM\SQLLIB
Ø -                    I
     Ø DB2PATH                        C:\Program Files\IBM\SQLLIB
Ø -                    E
     Ø DB2PORTRANGE                   6ØØØØ:6ØØØ3
Ø -                    I
```

```
     Ø DB2SYSTEM                       XXX
Ø -                    G
     Ø DB2TEMPDIR                      C:\PROGRA~1\IBM\SQLLIB\
Ø -                    E
```

I have ordered the rows by db_num and then by reg_var_name. To get it to look more like the output from the **db2set** command I would order by level, but you won't quite get the same order.

The above information is equivalent to the output I would obtain if I did a **>db2set -all** command:

```
[e] DB2PATH=C:\Program Files\IBM\SQLLIB
[i] DB2_DXX_PATHS_ALLOWED_WRITE=ANY
[i] DB2_DXX_PATHS_ALLOWED_READ=ANY
[i] DB2_DOCHOST=localhost
[i] DB2ACCOUNTNAME=XXX\db2admin
[i] DB2INSTOWNER=XXX
[i] DB2PORTRANGE=60000:60003
[i] DB2_RR_TO_RS=YES
[i] DB2INSTPROF=C:\PROGRA~1\IBM\SQLLIB
[i] DB2DBDFT=sample
[i] DB2COMM=TCPIP
[g] DB2_EXTSECURITY=NO
[g] DB2_DOCPORT=51000
[g] DB2_DOCHOST=localhost
[g] DB2SYSTEM=XXX
[g] DB2PATH=C:\Program Files\IBM\SQLLIB
[g] DB2INSTDEF=DB2
[g] DB2ADMINSERVER=DB2DAS00
```

One entry that I get using the UDF instead of the **db2set** command is the value for DB2TEMPDIR, which I could get by issuing a **>set** command.

What I like about these UDFs is that I can use SQL to obtain information that previously I had to obtain from system commands. This allows me to store the information in tables with a timestamp and thus I have a historical view of my system and what changed when. I don't have to write a program and maintain it, I just have to write some simple SQL and schedule that SQL to run regularly. Give them a go and see what they can do for you.

*C Leonard*
*Freelance Consultant (UK)*

Attunity has announced Version 4.8 of its Attunity Integration Suite (AIS), with particular enhancements to the Attunity STREAM software.

The new release provides a platform for Change Data Capture (CDC) as a foundation for efficient and real-time data integration. CDC is an approach to real-time data integration that enables the processing of changes to enterprise data sources. With STREAM, organizations can eliminate batch windows and increase the efficiency of ETL processing, reduce latency in delivering critical business information, reduce costs, and increase the ROI on existing and new data warehousing and business intelligence investments.

Version 4.8 provides a single solution supporting CDC for enterprise data sources, including DB2, VSAM, IMS, Adabase, and Oracle, and HP NonStop, AS/400, Unix, and Windows platforms.

For further information contact:
URL: www.attunity.com/content/PressRelease.Asp?PRName=20050927a.

* * *

OpenDemand Systems has announced Version 5.0 of OpenLoad, its browser-based, enterprise-functional, load-testing and monitoring solution for dynamic Web sites, applications, and services.

OpenLoad uses WebSphere and DB2 UDB. Its User Scenario Session Finder is an intelligent, script-free wizard that allows users to build dynamic data-driven tests without having to code and debug, or understand, the inner workings of the applications they are testing.

The product also allows users to proactively manage the performance of their critical Web applications from a single browser-based console. This new feature enables users to leverage real-world user scenarios developed for functional and load testing to continually monitor customer transactions and correlate them with their back-end systems to identify and fix problems before they impact Web site visitors.

For further information contact:
URL: www.opendemand.com/openload.

* * *

GridApp Systems has announced GridApp Clarity, which is a software solution that focuses on automating time-consuming database operations, while offering a single view across heterogeneous database platforms.

Clarity is intended to complement general-purpose data centre automation software, providing a best practices approach to cross-platform database management using specialized automation processes for configuration management, patch management, auditing/compliance, replication, and encryption. Clarity transforms the existing database infrastructure into a centrally administered and automated data utility, regardless of whether the platform is DB2, Oracle, SQL Server, Sybase, or a combination of these databases. By simplifying repetitive tasks, enforcing standard operating procedures, and correlating events for root cause analysis, organizations can improve database reliability, performance and cost efficiency.

For further information contact:
URL: www.gridapp.com/director.php/clarity.