133

DB2 update

November 2003

In this issue

DB2 Update

Published by

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

North American office

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

Subscriptions and back-issues

A year's subscription to *DB2 Update*, comprising twelve monthly issues, costs £255.00 in the UK; $380.00 in the USA and Canada; £261.00 in Europe; £267.00 in Australasia and Japan; and £265.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 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

Contributions

When Xephon is given copyright, articles published in *DB2 Update* are paid for at the rate of £100 ($160) per 1000 words and £50 ($80) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £20 ($32) 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.

Parse DISPLAY DATABASE output

Have you ever needed to look for all the DB2 databases containing tablespaces or indexspaces with a particular status that you want to do something about? Well, I have on several occasions, so I wrote a little REXX EXEC called GENDBCMD to capture and parse the output from the DISPLAY DATABASE command and generate commands. It uses commonly-issued DB2 commands like:

```
-DISPLAY DATABASE(*) SPACE(*) RES
```

It uses various qualifiers to filter the results based on what I am trying to find. The results of these commands almost always contain all the information necessary to issue additional commands. I'm sure everyone has issued commands from the console, DB2I or the SDSF ULOG, and scrolled through the results making notes in order to enter additional commands. This is a fact of life with DB2.

Occasionally, I find myself in a situation where I want to issue a bunch of similar commands based on the results of a single DB2 display command. Here's a recent example.

The DISPLAY DATABASE command has improved over the years to allow filtering of the output using the USE, CLAIMERS, LOCKS, LPL, WEPR, and RESTRICT keywords. We have a fairly large environment – three production sysplexes (12-way, 4-way, and 3-way) and three development sysplexes (3-way, 2-way, and 2-way), all using DB2 Datasharing. The 12-way is in one data centre and the rest are in another. This configuration allows us to recover in either direction. We also have an 'idle' recovery sysplex running Extended Remote Copy (XRC) to mirror DASD from the primary to the back-up data centre. Every three months we will test our DR plan. XRC is really nice in this situation, but there are still some DB2 recovery considerations. One of the tedious recovery tasks is cleaning up all the Group Recovery Pending (GRECP) restrictions at the recovery site. Sometimes there is a small number, sometimes there is a large number (in the hundreds). The simple solution is to issue a –START DATABASE against all the affected databases and their affected tablespaces and indexspaces. The following command will nicely identify the ‘hit’ list:

```
-DISPLAY DATABASE(*) SPACE(*) RES(GRECP) LIMIT(*)
```

In the results is all the information necessary to format the required:

```
-START DATABASE(xxxxxxx) SPACE(yyyyyy)
```

or:

```
-START DATABASE(xxxxxxx) SPACE(yyyyyy) PART(nnn)
```

Another example is putting a group of databases in ACCESS(RO), based on a condition/restrictions or just exploiting the already existing DB2 wildcarding capability in the DISPLAY, START, and STOP commands. GENDBCMD will generate all the individual commands, which could avoid some authorization issues with using wildcards.

My GENDBCMD EXEC will generate the commands for you. You can preview the generated commands (SIMULATE mode) or execute them inline (EXECUTE mode). The EXEC is intended to run in batch but could run interactively if you ALLOCATE the DB2 LOADLIB and issue a TSO TSOLIB command before entering ISPF or have the DB2 LOADLIB in your TSO PROC STEPLIB or in the Linklist.

The parameters for GENDBCMD are:

- SSID – the DB2 subsystem ID.
- DATABASE – the database (or pattern) to use in the driving DISPLAY command.
- SPACENAM – the space name (or pattern) to use in the driving DISPLAY.
- DISPOPT – the DISPLAY filter option desired for the DB2 DISPLAY command.
- MODE – execution mode (SIMULATE or EXECUTE).

- COMMAND – the DB2 command to generate (START, STOP, DISPLAY, or whatever).
- CMDSUFF – any parameters to add to the end of the generated command.

The DISPOPT can be any of the filtering options of the DB2 DISPLAY command like:

- USE
- CLAIMERS
- LOCKS
- LPL
- WEPR
- RESTRICT
- RESTRICT(GRECP) or any valid restriction or a list of restrictions.

The driving DISPLAY command is built with LIMIT(*). The generated commands are built by concatenating the supplied COMMAND with the database name, spacename, partition (if applicable), and any user-supplied command suffix (CMDSUFF).

The return code reasons are:

- 00 – everything worked.
- 04 – the DISPLAY command was truncated because of lack of internal DB2 space for command output or no databases found with the filter.
- 08 – syntax error found while trying to execute the driving DISPLAY command.
- 12 – DB2 not found/loaded (probably a STEPLIB or REGION error).

Here is some sample JCL to run GENDBCMD:

```
//jobname JOB  (ØØØØ),'GENDBCMD',MSGCLASS=T,REGION=ØM,NOTIFY=&SYSUID
```

```
/*JOBPARM SYSAFF=*
//*********************************************************************
//* RUNS THE GENDBCMD REXX EXEC                                       *
//*                                                                   *
//* PARMS: GENDBCMD   - REQUIRED - THE EXEC NAME                      *
//*        SSID       - REQUIRED - THE DB2 SSID DEF: DB2A             *
//*        DATABASE   - REQUIRED - THE DATABASE DEF: *                *
//*        SPACENAM   - REQUIRED - THE SPACE NAME DEF: *              *
//*        DISPOPT    - REQUIRED - THE DISPLAY OPTION DEF: RES(GRECP) *
//*        MODE       - REQUIRED - SIMULATE OR EXECUTE DEF: SIMULATE  *
//*        COMMAND    - REQUIRED - DB2 COMMAND DEF: START             *
//*        CMDSUFF    - OPTIONAL - ADDITIONAL PARMS FOR COMMAND       *
//*                                                                   *
//*********************************************************************
//GENDBCMD EXEC PGM=IKJEFTØ1,
//*
//*        PARM='GENDBCMD DB2A * * RES(GRECP) SIMULATE START'
//*        PARM='GENDBCMD DB2A * * RES(GRECP) EXECUTE START'
//*
//* UNCOMMENT THE LINES BELOW TO SEE OTHER EXAMPLES OF GENDBCMD
//*
//*        PARM='GENDBCMD DB2A * * RES SIMULATE DISPLAY'
//*        PARM='GENDBCMD DB2A * * RES(RECP,RBDP) EXECUTE DISPLAY'
//*        PARM='GENDBCMD DB2A * * RES(UTRO) SIMULATE DISPLAY'
//*        PARM='GENDBCMD DB2A * * RES(UTRW) EXECUTE DISPLAY'
//         PARM='GENDBCMD DB2A * * USE SIMULATE DISPLAY'
//*        PARM='GENDBCMD DB2A * * USE EXECUTE DISPLAY'
//*        PARM='GENDBCMD DB2A * * CLAIMERS SIMULATE DISPLAY'
//*        PARM='GENDBCMD DB2A * * LOCKS SIMULATE DISPLAY'
//*
//* UNCOMMENT THE LINES BELOW TO SEE USING STOP/START COMMANDS
//*
//*        PARM='GENDBCMD DB2A AAA* * USE SIMULATE STOP'
//*        PARM='GENDBCMD DB2A AAA* * USE SIMULATE START ACCESS(RO)'
//*
//* UNCOMMENT THE LINES BELOW TO SEE HOW GENDBCMD REACTS TO BAD PARMS
//*
//*        PARM='GENDBCMD DB2A * * GARBAGE SIMULATE DISPLAY'
//*        PARM='GENDBCMD DB2A * * RES(RW) EXECUTE DISPLAY'
//*
//STEPLIB  DD   DSN=db2.loadlib,DISP=SHR
//SYSEXEC  DD   DSN=your.rexx.pds,DISP=SHR
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   DUMMY
```

I didn't have any real conditions to run against to write this article, so all my test runs used a DISPLAY DB(*) SPACE(*) USE LIMIT(*) command. Here are some excerpts from output of a

SIMULATE run:

```
----- Command: -DIS DB(*) SPACE(*) USE LIMIT(*) Started: 12:31:39 ------

  SIMULATE  SIMULATE  SIMULATE  SIMULATE  SIMULATE  SIMULATE  SIMULATE
.
.
-DISPLAY DB(DSNDBØ6) SPACE(IXIVIEWD)
-DISPLAY DB(DSNDBØ6) SPACE(IXIVLSG)
-DISPLAY DB(DSNDBØ6) SPACE(IXIVLVL)
171 commands generated for database DSNDBØ6
Processing Database: AAADBØØØ
-DISPLAY DB(AAADBØØØ) SPACE(TAAAI)
-DISPLAY DB(AAADBØØØ) SPACE(TAAAP)
-DISPLAY DB(AAADBØØØ) SPACE(TAAAS)
-DISPLAY DB(AAADBØØØ) SPACE(TAACC)
-DISPLAY DB(AAADBØØØ) SPACE(TAACH)
-DISPLAY DB(AAADBØØØ) SPACE(TAACN)
-DISPLAY DB(AAADBØØØ) SPACE(TAAFA)
-DISPLAY DB(AAADBØØØ) SPACE(TAAFG) PART(ØØ1)
-DISPLAY DB(AAADBØØØ) SPACE(TAAFG) PART(ØØ2)
-DISPLAY DB(AAADBØØØ) SPACE(TAAFG) PART(ØØ3)
-DISPLAY DB(AAADBØØØ) SPACE(TAAFG) PART(ØØ3)
.
.
-DISPLAY DB(CCCDBØØ1) SPACE(IXAPJ4)
-DISPLAY DB(CCCDBØØ1) SPACE(IXARN1)
15 commands generated for database CCCDBØØ1
Processing Database: DDDDBØØ1
-DISPLAY DB(DDDDBØØ1) SPACE(TAACU) PART(ØØ1)
-DISPLAY DB(DDDDBØØ1) SPACE(TAACU) PART(ØØ2)
2 commands generated for database DDDDBØØ1

List is incomplete: DSNT3Ø6I  -DB2A RESPONSE MESSAGE LIMIT HAS BEEN
REACHED

33 Databases found 1961 commands generated

  SIMULATE  SIMULATE  SIMULATE  SIMULATE  SIMULATE  SIMULATE  SIMULATE

--- Command: -DIS DB(*) SPACE(*) USE LIMIT(*) Completed: 12:31:4Ø ------

GENDBCMD completed RC=4 elapsed time 3.323641
```

Here are some excerpts from output of an EXECUTE run generating DISPLAY command:

```
------Command: -DIS DB(*) SPACE(*) USE LIMIT(*) Started: 11:13:22 ------
```

```
  EXECUTE  EXECUTE  EXECUTE  EXECUTE  EXECUTE  EXECUTE  EXECUTE
.
.
-- -DISPLAY DB(AAADBØØØ) SPACE(TAAFG) PART(ØØ5)  Started: 11:13:38 -----
DSNT36ØI  -DB2A ***********************************
DSNT361I  -DB2A *  DISPLAY DATABASE SUMMARY
                   GLOBAL
DSNT36ØI  -DB2A ***********************************
DSNT362I  -DB2A     DATABASE = AAADBØØØ  STATUS = RW
                    DBD LENGTH = 969Ø2

DSNT397I  -DB2A
NAME     TYPE PART STATUS            PHYERRLO PHYERRHI CATALOG  PIECE
-------- ---- ---- ----------------- -------- -------- -------- -----
TSAFG    TS    ØØ5 RW
******* DISPLAY OF DATABASE AAADBØØØ ENDED      *********************
DSN9Ø22I  -DB2A DSNTDDIS 'DISPLAY DATABASE' NORMAL
COMPLETION
-- -DISPLAY DB(AAADBØØØ) SPACE(TAAFG) PART(ØØ5)  Completed: 11:13:38 ---
-------- -DISPLAY DB(AAADBØØØ) SPACE(TAAHY)  Started: 11:13:38 ---------
DSNT36ØI  -DB2A ***********************************
DSNT361I  -DB2A *  DISPLAY DATABASE SUMMARY
                   GLOBAL
DSNT36ØI  -DB2A ***********************************
DSNT362I  -DB2A     DATABASE = AAADBØØØ  STATUS = RW
                    DBD LENGTH = 969Ø2

DSNT397I  -DB2A
NAME     TYPE PART STATUS            PHYERRLO PHYERRHI CATALOG  PIECE
-------- ---- ---- ----------------- -------- -------- -------- -----
TSAHY    TS        RW

******* DISPLAY OF DATABASE AAADBØØØ ENDED      *********************
DSN9Ø22I  -DB2A DSNTDDIS 'DISPLAY DATABASE' NORMAL
COMPLETION
------- -DISPLAY DB(AAADBØØØ) SPACE(TAAHY)  Completed: 11:13:38 -------
.
.

146 commands generated for database XXXDBØØ1

List is incomplete: DSNT3Ø6I  -DB2A RESPONSE MESSAGE LIMIT HAS BEEN
REACHED

33 Databases found 2Ø58 commands generated

  EXECUTE  EXECUTE  EXECUTE  EXECUTE  EXECUTE  EXECUTE  EXECUTE
----Command: -DIS DB(*) SPACE(*) USE LIMIT(*) Completed: 11:16:16 ------

GENDBCMD completed RC=4 elapsed time 3.323641
```

Here is the GENDBCMD EXEC:

```
/*********************************************************************/
/*                               REXX                                */
/*********************************************************************/
/* Purpose: Generate DB2 commands from DISPLAY DB command output     */
/*-------------------------------------------------------------------*/
/* Syntax:  GENDBCMD ssid database spacenam dispopt mode command     */
/*-------------------------------------------------------------------*/
/* Parms: SSID       - DB2 SSID                                      */
/*        DATABASE   - DB2 database(s) to display (default: *)       */
/*        SPACENAM   - DB2 space names to display (default: *)       */
/*        DISPOPT    - DB2 restriction to display ie RES(GRECP)      */
/*        MODE       - SIMULATE or EXECUTE                           */
/*        COMMAND    - DB2 Command to generate (START, STOP or DIS)  */
/*        CMDSUFF    - Any additional parms for generated command    */
/*                                                                   */
/*********************************************************************/
/* Housekeeping                                                      */
/*********************************************************************/
 parse upper source . . execname .
 started = time('r')
/*********************************************************************/
/* Establish counters and EXITRC                                     */
/*********************************************************************/
 dbcnt  = Ø
 spcnt  = Ø
 totcnt = Ø
 EXITRC = Ø
/*********************************************************************/
/* Accept parms if defaults are not desired                          */
/*********************************************************************/
 arg ssid database spacenam dispopt mode command cmdsuff
 if ssid     = '' then ssid     = 'DB2A'
 if database = '' then database = '*'
 if spacenam = '' then spacenam = '*'
 if dispopt  = '' then dispopt  = 'RES(GRECP)'
 if mode     = '' then mode     = 'SIMULATE'
 if command  = '' then command  = 'START'
/*********************************************************************/
/* Build the DB2 command from user input or defaults                 */
/*********************************************************************/
 display = '-DIS DB('database') SPACE('spacenam')' dispopt 'LIMIT(*)'
/*********************************************************************/
/* Execute the DISPLAY command                                       */
/*********************************************************************/
 call db2cmd 'HIDE' display
/*********************************************************************/
/* Start of results                                                  */
```

```
/*********************************************************************/
 say center(' Command:' display 'Started:' time()' ',78,'-')
 say
 say center(copies(' 'mode' ',8),78)
 say
/*********************************************************************/
/* Loop through results                                              */
/*********************************************************************/
 do i=1 to result.Ø
    parse var result.i word1 word2 word3 word4 word5 word6 word7
    select
/*********************************************************************/
/* Everything to throw away                                          */
/*********************************************************************/
      when word1 = 'DSNT36ØI' then nop
      when word1 = 'DSNT361I' then nop
      when word1 = 'DSNT397I' then nop
      when word1 = 'DSN9Ø23I' then nop
      when word1 = '--------' then nop
      when word1 = '*'        then nop
      when word1 = '-'        then nop
      when word1 = '*******'  then nop
      when word1 = 'DBD'      then nop
      when word1 = 'NAME'     then nop
/*********************************************************************/
/* No databases found                                                */
/*********************************************************************/
      when word1 = 'DSNT365I' | word1 = 'DSNT367I' then
           do
            say 'No databases found in' dispopt 'for DB('database')',
                'SPACE('spacenam')'
            EXITRC = 4
            signal shutdown
           end
/*********************************************************************/
/* Bad keyword or value for DISPOPT                                  */
/*********************************************************************/
      when word1 = 'DSN9Ø15I' | word1 = 'DSN9ØØ1I' then
           do
            say 'Parm error:' result.i
            EXITRC = 8
            signal shutdown
           end
/*********************************************************************/
/* Get the database name                                             */
/*********************************************************************/
      when word1 = 'DSNT362I' then
           do
            call dbsumm
```

```
/*********************************************************************/
/* Begin processing the database                                     */
/*********************************************************************/
            db = word5
            dbcnt = dbcnt + 1
            say 'Processing Database:' db
            spcnt = Ø
           end
/*********************************************************************/
/* Good end of list                                                  */
/*********************************************************************/
      when word1 = 'DSN9Ø22I' then call dbsumm
/*********************************************************************/
/* Bad end of list                                                   */
/*********************************************************************/
      when word1 = 'DSNT3Ø6I' then
           do
            call dbsumm
            say
            say 'List is incomplete:' result.i
            EXITRC = 4
           end
/*********************************************************************/
/* Get the tablespaces, indexspaces and partitions                   */
/*********************************************************************/
      otherwise
           do
            snam = word1
            part = substr(result.i,16,3)
/*********************************************************************/
/* Determine if a partition was found                                */
/*********************************************************************/
            if part = '   ' then
               piece = 'DB('db') SPACE('snam')'
            else
               piece = 'DB('db') SPACE('snam') PART('part')'
/*********************************************************************/
/* Assemble the DB2 command (add the '-' if missing)                 */
/*********************************************************************/
            if substr(command,1,1) = '-' then
               dbcmd = command piece cmdsuff
            else
               dbcmd = '-'command piece cmdsuff
/*********************************************************************/
/* Determine if this is SIMULATE or EXECUTE mode                     */
/*********************************************************************/
            if mode = 'EXECUTE' then
               call db2cmd 'SHOW' dbcmd
            else
```

```
               say dbcmd
            spcnt = spcnt + 1
           end
    end
 end
/*********************************************************************/
/* Shutdown                                                          */
/*********************************************************************/
 shutdown: ended = time('e')
           elapsed = ended - started
/*********************************************************************/
/* Display status                                                    */
/*********************************************************************/
           say
           say dbcnt 'Databases found' totcnt 'commands generated'
           say
           say center(copies(' 'mode' ',8),78)
/*********************************************************************/
/* End of results                                                    */
/*********************************************************************/
           say
           say center(' Command:' display 'Completed:' time()' ',78,'-')
           say
           say execname 'completed RC='EXITRC 'elapsed time' elapsed
/*********************************************************************/
/* Exit with a return code                                           */
/*********************************************************************/
           exit(EXITRC)
/*********************************************************************/
/* Queue and execute the DB2 command                                 */
/*********************************************************************/
 db2cmd: arg option execcmd
         drop output.
         if option = 'HIDE' then
            call outtrap 'result.'
         else
            call outtrap 'output.'
         queue execcmd
         queue "END"
        "DSN SYSTEM("ssid")"
/*********************************************************************/
/* Error calling DB2                                                 */
/*********************************************************************/
         if result.Ø = Ø then
            do
             EXITRC = 12
             say 'DB2 modules not found, confirm STEPLIB is correct'
             signal shutdown
            end
```

```
/*********************************************************************/
/* Get rid of the garbage lines on the stack                         */
/*********************************************************************/
         pull emptystack
         pull emptystack
/*********************************************************************/
/* Display the command output if this was not a HIDE request         */
/*********************************************************************/
         if option <> 'HIDE' then
            do
             say center(' 'execcmd 'Started:' time()' ',78,'-')
             do j=1 to output.Ø
                say output.j
             end
             say center(' 'execcmd 'Completed:' time()' ',78,'-')
            end
         return
/*********************************************************************/
/* Produce the database summary and totals                           */
/*********************************************************************/
 dbsumm: if spcnt > Ø then
            do
             say spcnt 'commands generated for database' db
            end
         totcnt = totcnt + spcnt
         return
```

Robert Zenuk
Systems Programmer (USA)

The DB2AUDIT facility

What does the DB2AUDIT function give you? Well, it's an auditing tool that lets you see who is accessing your data, who is updating it, and who has tried to access data to which they have no access authority! You audit at the instance level (not just at the database level), but you can audit all databases within an instance.

The statements in this article were run on a Windows 2000 system using the db2admin userid (install userid) running DB2 UDB 8.1 FP1.

Who can issue the DB2AUDIT commands? You need to be SYSADM to use the DB2AUDIT commands.

Which manual contains a description of the audit function? It is the *Administration Guide – Implementation* manual (SC09-4820-00).

So what do I have to do? The basic steps of what you need to do are:

- Configure the audit facility.
- Start the audit facility.
- Let the users onto the system.
- Extract data from the audit log.
- Stop the audit facility.

You configure the audit facility using the db2audit configure command. This creates the db2audit.cfg file. When you start the audit facility, you create the db2audit.log file, and when you extract data from the audit log using the db2audit extract command, you create the <*event*>.del files. All these steps are shown in the example below.

What is an audit 'event'? The audit facility can monitor things such as who created a table, who is selecting from a table etc. These tasks of creating, selecting etc, are called 'events'.

Which events can the audit facility monitor? The audit facility can monitor many events, and these are all fully listed in the *Administration Guide – Implementation* manual, p272. Events are grouped into the following seven categories:

- Audit – AUDIT
- Authorization checking – CHECKING
- Object maintenance – OBJMAINT
- Security maintenance – SECMAINT

- System administration – SYSADMIN
- User validation – VALIDATE
- Operation context – CONTEXT

Some of the major categories from the above list are:

- Authorization checking (CHECKING) – who is accessing what tables.
- Object maintenance (OBJMAINT) – when objects are dropped.
- Security maintenance (SECMAINT) – who is granting what to whom.

The table names (in capitals) in the list above are the names of the tables (in the C:\Program Files\IBM\SQLLIB\<*instance*>\security directory) that will be created when you run the db2audit extract command.

What is the layout of the category tables? The layout of the category tables is given in the *Administration Guide – Implementation* manual in Chapter 5.

Are the category tables overwritten every time you run the db2audit extract command? No they are not – however, you cannot append to the tables, you have to delete them before you issue the db2audit extract command again. If you do try to issue the command again without deleting the tables, you will get an AUD0021N error message.

Can I monitor successes and failures? Yes, you can monitor event failures, event successes, or both.

Where are the audit control files? There are two audit files: the configuration file (db2audit.cfg) and the output file (db2audit.log). They both reside in the C:\Program Files\IBM\SQLLIB\<*instance*>\security directory. The db2audit.cfg file gets created when you issue the db2audit configure command. The db2audit.log file gets created when you start auditing.

Is there an impact on performance if I run the audit facility? Yes. You can either have the audit facility record in the logs synchronously or asynchronously in relation to when the triggering event occurs. You use the DBM parameter audit_buz_sz to determine this. Basically, a value of 0 means write synchronously, and a value greater than 0 means buffer up the writes. The default value is 0.

```
>db2 get dbm cfg | find /i "audit"
 Audit buffer size (4KB)                  (AUDIT_BUF_SZ) = Ø
```

There is obviously a trade-off between having synchronous and asynchronous writes, so you will need to do tests to determine the optimum size for AUDIT_BUF_SZ, for your system, assuming that you are willing to have audit information buffered.

What are the different db2audit commands? Below is a summary of the various db2audit commands:

- >db2audit configure – populates the configuration file.
- >db2audit extract – extracts data from the audit log.
- >db2audit start – starts the audit process.
- >db2audit stop – stops the audit process.
- >db2audit describe – lists what categories you are monitoring.
- >db2audit flush – flushes the audit_buz_sz buffer.
- >db2audit prune – deletes records from the audit log.

For help on all the db2audit command use: >db2audit?

How do I query the audit log? You use the extract command to query the audit log. The basic form of the command is:

```
>db2audit extract delasc delimiter ! database <db-alias>
```

This will extract data for all categories for the <*db-alias*> database into the category files, using the exclamation mark (!) as a field delimiter. This feature is very useful when you want to import the data from the tables into a spreadsheet.

If you just want to extract information for one category, the command would be in the form:

```
>db2audit extract delasc delimiter ! category audit database <db-alias>
```

When you run the extract command, the following tables are created in the C:\Program Files\IBM\SQLLIB\<*instance*>\security directory:

- audit.del
- checking.del
- context.del
- objmaint.del
- secmaint.del
- sysadmin.del
- validate.del.

The filenames correspond to the different events that the audit facility can monitor. I will show later which files are populated by which tasks.

So let's look at an example. Suppose we want to create audit records for every possible auditable event. Remember that this is at the instance level. We will be using the SAMPLE database. We would issue the command:

```
>db2audit configure scope all status both
AUDØØØØI  Operation succeeded.
```

Check what we have set using the command:

```
>db2audit describe
DB2 AUDIT SETTINGS:
Audit active: "TRUE "
Log errors: "TRUE "
Log success: "TRUE "
Log audit events: "TRUE "
Log checking events: "TRUE "
Log object maintenance events: "TRUE "
Log security maintenance events: "TRUE "
Log system administrator events: "TRUE "
Log validate events: "TRUE "
Log context events: "TRUE "
Return SQLCA on audit error: "FALSE "
```

If we are happy with the above settings, we can start the audit facility using:

```
>db2audit start
AUDØØØØI  Operation succeeded.
```

Now create a test table in our SAMPLE database:

```
>db2 connect to sample
>db2 create table helØ1 (id int)
```

Let's flush the buffer, just to make sure we have all the audit data:

```
>db2audit flush
```

Run the audit extract command:

```
>db2audit extract delasc delimiter ! database sample
```

The following audit files would have been populated:

- checking.del, three rows
- context.del, two rows
- objmaint.del, one row
- secmaint.del, one row
- audit.del   , empty
- sysadmin.del   , empty
- validate.del, empty.

The context.del file shows me the DDL that was issued to create the table, and who issued the command. The objmaint.del shows me that a table was created, the table name, and who issued the command, but not the DDL used. You can see that, for a simple create table statement, seven rows were written to the audit files.

Now insert a row into the test table:

```
>db2 insert into helØ1 values(8)
```

Run the audit flush and extract commands:

```
>db2audit flush
>db2audit extract delasc delimiter ! database sample
```

The following audit files would have been populated:

- checking.del, one row
- context.del, four rows
- audit.del, empty
- objmaint.del, empty
- secmaint.del, empty
- sysadmin.del, empty
- validate.del, empty.

The context.del file shows me the SQL that was issued to insert the row into the table, and who issued the command.

Now update the row in the test table:

```
>db2 update helØ1 set id = 4
```

Run the audit flush and extract commands:

```
>db2audit flush
>db2audit extract delasc delimiter ! database sample
```

The same audit files would have been populated as for the insert command. The context.del file shows me the SQL that was issued to update the row in the table, and who issued the command.

Now select from our test table:

```
>db2 select * from helØ1
```

Run the audit flush and extract commands:

```
>db2audit flush
>db2audit extract delasc delimiter ! database sample
```

The same audit files have been populated as for the insert command. The context.del (five rows) file shows me the SQL that was issued to select from the table, and who issued the command.

Stop auditing the system:

```
>db2audit stop
AUDØØØØI  Operation succeeded.
```

As you can see, it is only the checking.del and context.del files that are populated when inserting/updating/selecting records, with anywhere between four and five records being written for the operation. So the number of SQL operations that are performed on your data will determine the rate of growth of these files. This must be carefully monitored.

We have now completed the example, so here are a few more questions and answers.

Can I delete records from the audit log? Yes, you can – you use the >db2audit prune command. There are three possible parameters to this command – all, a date/time stamp, and a pathname to a directory. The all parameter is fairly obvious! You are left with one record in the audit log (recording that you issued the prune all command). The date/time stamp deletes all records that have a timestamp on or before the date/time stamp specified. You can also specify a pathname because the prune command is also logged, and if the disk on which the log is is full, then what do you do? The pathname parameter allows you to specify a temporary path where the prune command will be logged. I haven't tested this parameter.

Can I delete records from the seven audit files? No – you have to delete all the files every time before you run the >db2audit extract command.

What is the default audit configuration? The default configuration is that you will collect data only for failures. Some other possible configurations are shown below:

- To create audit records for every possible auditable event (as in our example above) configure the audit facility as follows:

  ```
  >db2audit configure scope all status both
  ```

- If you wanted to audit only those events that fail, configure the audit facility as follows:

  ```
  >db2audit configure scope all status failure
  ```

- If you wanted to create audit records for successful grants on objects, configure the audit facility as follows:

```
>db2audit configure scope checking status success
```

Can I list out only event failures for a certain database in my instance? Yes, you can – you would use the following extract command:

```
>db2audit extract delasc delimiter ! database <db-alias> status failure
```

(and to list out just successes, simply replace the word ‘failure’ with the word ‘success’). Obviously, you cannot list out failures if you are only collecting successes and *vice versa*!

Where do I find DB2AUDIT error messages? These can be found on p281 of the *Administration Guide: Implementation* (search for SQL1322N).

I hope I have shown how easy it is to set up the DB2AUDIT facility, and how you can extract useful data from the audit logs.

C Leonard
Freelance Consultant (UK)

How DB2 supports and exploits Web services technologies – part 2

This month we conclude the article looking at Web services and DB2.

DB2 AS A WEB SERVICE PROVIDER

DB2 data and applications (stored procedures) can be accessed as Web services by developing Web service implementations, which could be Java classes or EJB components that access DB2 through JDBC requests. However, each time a different type of data access is desired, a new Web service implementation would be needed. A simpler, and more generic, alternative is to use the support shipped in DB2 V8.1 that processes DADX (Document Access Definition extension) files in conjunction with a general SOAP-enabled server, such as WebSphere. A DADX file is an XML document that represents a DB2 Web service.

The DADX file supports three types of operation for making database requests as Web services. The relational results from these requests are tagged as XML by the Web services Object Runtime Framework (WORF) in a customizable way. The three types of operation are:

1 SQL Request – any SQL request can be issued, including SELECT, UPDATE, user-defined functions, and DB2 XML Extender column facilities.

2 Stored procedure call – any DB2 stored procedure can be invoked.

3 DB2 XML Extender stored procedure call – any DB2 XML Extender stored procedure can be invoked. These stored procedures can provide two types of functionality. First, composition is used for detailed control on the tagging of the generated document using a DAD (Document Access

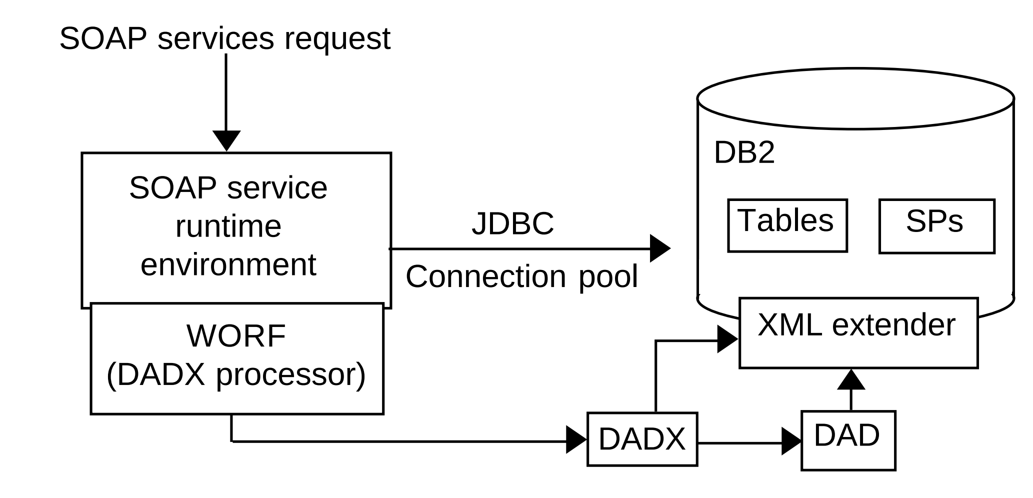

Figure 1: WORF architecture

Definition). This approach can be used to control where element repetition takes place or the number of documents produced, for example. Second, decomposition is used to ‘shred’ an XML document and transform the result into transactional data.

The DADX can be created using a text or XML editor or with Websphere Studio Application developer. To deploy a DADX-based Web service, the required DADX files are placed in the appropriate SOAP-enabled Web server directory along with the WORF.

WEB SERVICES OBJECT RUNTIME FRAMEWORK

WORF provides the runtime environment for Web services defined by DADX. WORF is implemented as an extension to an Apache SOAP 2.2 run-time component. It runs with the Apache Web server and Websphere application server.

Figure 1 shows WORF hosted in a SOAP service run-time environment. WORF receives an HTTP SOAP GET or POST service request from the SOAP RPC router. The service endpoint of the request specifies a DADX or DTD (Document Type Definition) file and the requested action. WORF converts DTDs to XSDs. DB2 uses either DTDs or XSDs, but WSDL uses only XSDs. WORF loads the DADX file specified in the request and connects to DB2 to run any SQL statements. Parameter markers in SQL statements are replaced with the requested values. WORF formats the result of the SQL statement into XML, converting types as necessary, and returns a SOAP response envelope to the SOAP client.

Generating WSDL and XML schemata from the DADX

The WORF produces WSDL and XML schemata for the operations included in the DADX. The Web services client sees operations and parameters only.

Testing DADX-based Web services through WORF

WORF provides a facility to test DADX files and provides support for connection pooling and security management.

TOOLS FOR DB2-BASED WEB SERVICES

Many tools in the market have kept pace with the rapid emergence of Web services; both the IBM Websphere Studio product family and Microsoft Visual Studio with its .Net include extensive general Web service development support. Web Sphere studio, however, has specific support for DB2 services. Websphere studio simplifies the task of developing DB2 Web services; a user can build a Web service without writing any code. It also provides a rich set of functions for building the applications that access Web services. The latest support includes the automatic creation of Java proxies from WSDL files, and in the future the support will be enhanced to generate SQL-bodied UDFs as described earlier in the article. The SQL UDF generator allows developers to specify the WSDL files of interest. The operations defined in a WSDL file can then be mapped to UDFs. The input and output of the operations are mapped to parameters and return values of the generated UDFs. In addition, two styles of UDF proxy are supported. The first is an early-binding form with the UDF mapped to a specific end-point. In this case, when the UDF is executed, one specific instance of a Web service is invoked and its results returned by the UDF. The second is a late binding form with the endpoints passed in at runtime. In this case, the UDF can be invoked against a set of Web services that support the same interface, and the results are returned in table form. Both the command line and wizard form of the tools generate SOAP UDFs with no coding required by developers.

CONCLUSION AND FUTURE RESEARCH

In this article, we have shown how a relational database such as DB2 can both support and exploit Web services technologies, to simplify access to data and stored procedures and to extend the reach of DB2 queries into external sources addressable through Web services. Using the inherent power of the DB2 engine, we can easily perform set-oriented queries over external data and enrich traditional SQL queries with real-time data. Although DB2 runs SOAP requests in a reliable and scalable environment, the fact that a database query depends on an external, potentially unreliable, low-bandwidth SOAP provider sets some limits on the overall query performance. IBM DB2 is working to extend the Web services support and Web service support in DB2 continues to evolve and mature.

Vikas Baruah
Senior Technical Specialist
American Management Systems (USA)

DB2 ZPARM tool

The major function of this REXX tool is to assist in the management of your shop's ZPARM and DECP source. Preventing regression is an important issue, and this tool allows you to compare your DSNTIJUZ source with your current DB2 system parameter settings before making any changes. Also, you can use this tool to regenerate ZPARM source based on the currently-running DB2 and compare system resource definitions between two DB2 subsystems.

Some third-party vendor products can also display ZPARM source, based on a running DB2. One such product is BMC catalog manager, but its format is not compatible with DSNTIJUZ, and some fields, such as SMFSTAT=10111000000000000000000000000000, are not further refined – it should be SMFSTAT = (1,3,4,5) or SMFSTAT = YES. Those are all taken care of in this tool and the generated ZPARM is sorted in alphabetical order.

PROCEDURE

The first option of this tool will ask for the DB2 subsystem name and type to be input, then it calls the DB2 supplied stored procedure DSNWZP to extract the current settings from the running DB2. It compares them with the source DSNTIJUZ job provided by the user and reports any differences in an ISPF dialog. If you want to know the meaning of a specific parameter, simply type S beside the row and a pop-up window will show you.

The second option will generate two members – dsnzxxxx for ZPARM, dsnhxxxx for DSNHDECP – in dataset HLQ.ZPARM.CNTL (HLQ is your TSO userid, xxxx is DB2 subsystem name).

When IBM maintenance brings a new parameter into ZPARM, my tool will report 'This parm is new'. Simply change the member default by adding this new parameter.

INSTALLATION

Follow this checklist for installation:

- Allocate a PDS dataset DB2.ZPARM.LIB; copy all members into it.
- Set up program DSN8ED7 by running IVP job DSNTEJ6Z.
- Copy member zparm to one of your TSO log-on procedures' SYSEXEC concatenations.
- Customize panel member regen1 and compare with the DB2 and MVS system name in your shop. This is to make sure you run this tool against a DB2 under the same MVS system.
- In a TSO command line, type zparm to invoke the tool.

The test environment is DB2 Version 7 in Z/OS 1.3.

Here is the code:

ZPARM.REX

```
/*********************************************************************/
/*                                                                   */
/*    REXX NAME: DSNZPARM                                            */
/*                                                                    */
/*        AUTHOR: LIJUN GAO                                           */
/*                                                                    */
/*       PURPOSE: INITIALIZE WORK FILE                                */
/*                                                                    */
/*                                                                    */
/**********************************************************************/
MSG_STATUS = MSG("OFF")
HLQ = SYSVAR(SYSUID)

"DELETE '"HLQ".DELETE.LISTCAT'"
"ALLOC F(HOLDIT)  DA('"HLQ".DELETE.LISTCAT') NEW",
   "SPACE(1,1) TRACKS DIR(Ø) UNIT(SYSDA) DSORG(PS) RECFM(V B)",
   "LRECL(125) BLKSIZE(629)"

"LISTC ENTRIES('"HLQ".ZPARM.WORK1') OUTFILE(HOLDIT)"
  IF  RC > Ø THEN DO
    "ALLOC F(WORK1)  DA('"HLQ".ZPARM.WORK1') NEW",
       "SPACE(14,1) TRACKS UNIT(SYSDA) CATALOG DSORG(PS) RECFM(F B)",
       "LRECL(8Ø)"
  END

"LISTC ENTRIES('"HLQ".ZPARM.WORK2') OUTFILE(HOLDIT)"
  IF  RC > Ø THEN DO
    "ALLOC F(WORK2)  DA('"HLQ".ZPARM.WORK2') NEW",
    "SPACE(14,1) TRACKS UNIT(SYSDA) CATALOG DSORG(PS) RECFM(F B)",
    "LRECL(8Ø)"
  END
"ALLOCATE DDNAME(SYSUEXEC) DSN('DB2.ZPARM.LIB') SHR REUSE"
"ALTLIB ACTIVATE USER(EXEC)"
"FREE FI(HOLDIT)"
"FREE FI(WORK1)"
"FREE FI(WORK2)"
"DELETE '"HLQ".DELETE.LISTCAT'"
"ALLOCATE DDNAME(SYSUEXEC) DSN('DB2.ZPARM.LIB') SHR REUSE"
"ALTLIB ACTIVATE USER(EXEC)"
ADDRESS ISPEXEC
"LIBDEF ISPPLIB DATASET ID('DB2.ZPARM.LIB')"
"LIBDEF ISPMLIB DATASET ID('DB2.ZPARM.LIB')"
"ISPEXEC SELECT PANEL(MAIN)"
"LIBDEF ISPPLIB"
"LIBDEF ISPMLIB"
ADDRESS TSO
"ALTLIB DEACTIVATE USER(EXEC)"
"FREE FI(WORK1)"
"FREE FI(WORK2)"
"FREE FI(INDD)"
"FREE FI(INDD3)"
"FREE FI(SYSUEXEC)"
```

COMPARE1.PAN

```
)PANEL KEYLIST(ISRSPBC,ISR)
)ATTR
! TYPE(TEXT) INTENS(LOW) SKIP(ON) COLOR(YELLOW)
# TYPE(TEXT) INTENS(HIGH) SKIP(ON) COLOR(TURQ)
~ TYPE(TEXT) INTENS(HIGH) SKIP(ON) COLOR(PINK)
¬ TYPE(INPUT) COLOR(GREEN) HILITE(USCORE)
* TYPE(INPUT) COLOR(GREEN)
_ TYPE(INPUT) INTENS(LOW) COLOR(TURQ)
)BODY
#----------------------------~COMPARE PANEL#---------------------------
+COMMAND =====>*ZCMD                                 +Scroll ===> ¬SCRO#
%
!DB2 subsystem name  . . .&DSSN
+(Specify the DB2 subsystem name like dsn3, dsn7 etc)
%
%
!DB2 Subsystem type  . . .&DTYP
+(Specify the DB2 subsystem type like test, quco or prod)
%
%
!Source zparm dataset name: DB2.&DTYP..&DSSN..SDSNSAMP.R71Ø(DSNTIJUZ)
+(Press enter if that's correct, otherwise input your override)
%
!Source zparm dataset name:_dsnam                                      #
+(Don't put '' on your overide dataset name)
%
! Zparm only:_z+(N: both zparm and dsnhdecp   Y: zparm only)
%
#
)INIT
.ZVARS = '(ZPOL)'
&ZPOL = N
&SCRO = DATA
&DSNAM = ''
)REINIT
.ZVARS = '(ZPOL)'
&ZPOL = N
&SCRO = DATA
&DSNAM = ''
)PROC
VPUT DSNAM SHARED
VPUT ZPOL SHARED
&ZSEL='CMD(ZPAMCOMP)'
)END
```

COMPOUT.PAN

```
)PANEL KEYLIST(ISRSPBC,ISR)
)ATTR
! TYPE(TEXT) INTENS(LOW) SKIP(ON) COLOR(YELLOW)
# TYPE(TEXT) INTENS(HIGH) SKIP(ON) COLOR(TURQ)
~ TYPE(TEXT) INTENS(HIGH) SKIP(ON) COLOR(PINK)
¬ TYPE(INPUT) COLOR(GREEN) HILITE(USCORE)
* TYPE(INPUT) COLOR(GREEN)
_ TYPE(INPUT) INTENS(LOW) COLOR(TURQ)
@ TYPE(OUTPUT) INTENS(HIGH) COLOR(YELLOW)
)BODY
#---------------------------~COMPARE PANEL#---------------------------
+COMMAND =====>*ZCMD                                  +Scroll ===> ¬SCRO#
%
%
%
+     Macro Name      Parameter      Current Setting      Source Value
+     ---------------------------------------------------------------
)MODEL
     @LMMACRO        @LMPARM        @LMVALUE            @SRVALUE
)INIT
&SCRO = DATA
.CURSOR = ZCMD
)PROC
)END
```

CVTVALUE.REX

```
/*
|| =================================================================
|| ROUTINE:   CVTVALUE          : CONVERT ZPARM VALUE TO COMPATIBLE
|| ARGUMENTS: LM_VALUE,LM_PARM : PARAMETER NAME AND ITS VALUE
|| RETURN:    CONV_VALUE        : CONVERTED VALUE
|| =================================================================
*/
  ARG INPUT_VALUE,INPUT_PARM
  IF INPUT_PARM = DEALLCT THEN DO
     MINUTES = ABS(SUBSTR(INPUT_VALUE,1,5))
     SECONDS = ABS(SUBSTR(INPUT_VALUE,7,5))
     IF MINUTES = Ø THEN MINUTES =''
     IF SECONDS = Ø THEN SECONDS =''
     CONV_VALUE = '('||MINUTES||','||SECONDS||')'
     IF MINUTES = '' & SECONDS = '' THEN CONV_VALUE = '(Ø)'
     IF MINUTES = 144Ø & SECONDS = '' THEN CONV_VALUE = NOLIMIT
  END
  ELSE DO
     FOUND = Ø
     CONV_VALUE = '('
     DO K = 1 TO 32
        A.K = SUBSTR(INPUT_VALUE,K,1)
        IF A.K =1 THEN DO
        FOUND = 1
        CONV_VALUE = CONV_VALUE||K||','
        END
     END
     IF FOUND = Ø THEN CONV_VALUE = '(Ø)'
     ELSE DO
     CONV_VALUE = SUBSTR(CONV_VALUE,1,(LASTPOS(',',CONV_VALUE)-1))
     CONV_VALUE = CONV_VALUE||')'
     END
  END
 RETURN CONV_VALUE
```

DEFAULT.REX

```
               ALPOOLX=32256,
               DBCHK=NO,
               DISABSCL=NO,
               DB2SUPLD=NO,
               IXQTY=Ø,
               MINRBLK=1,
               MXQBCE=32767,
               MXTBJOIN=15,
               OFFLOAD=YES,
               OPTCCOS1=OFF,
               OPTSUBQ1=NO,
               PKGLDTOL=NO,
               PTASKROL=YES,
               SARGSWRP=NO,
               SMSDCFL=,
               SMSDCIX=,
               TABLES_JOINED_THRESHOLD=16,
               TRACLOC=16,
               TSQTY=Ø,
               TWOBSDS=2,
               WRTHRSH=2Ø,
```

DONE.PAN

```
)PANEL KEYLIST(ISRSPBC,ISR)
)ATTR
! TYPE(TEXT) INTENS(LOW) SKIP(ON) COLOR(YELLOW)
# TYPE(TEXT) INTENS(HIGH) SKIP(ON) COLOR(TURQ)
~ TYPE(TEXT) INTENS(HIGH) SKIP(ON) COLOR(PINK)
¬ TYPE(INPUT) COLOR(GREEN) HILITE(USCORE)
* TYPE(INPUT) COLOR(GREEN)
```

```
_ TYPE(INPUT) INTENS(LOW) COLOR(TURQ)
)BODY
#-----------------------------~Source Generation Output#----------------
+COMMAND =====>_ZCMD                                 +Scroll ===> ¬SCRO#
%
%
!DB2 subsystem name  . . .&DSSN#
+(Specify the DB2 subsystem name like dsn3, dsn7 etc)
%
%
%
! The zparm source has been generated in &output
%
#
)INIT
)PROC
VGET OUTPUT SHARED
VGET DSSN SHARED
&ZSEL='PANEL(MAIN)'
)END
```

DSNG00.MSG.MSG

```
DSNGØØ1 'Invalid Name'                 .HELP=*   .ALARM=YES
'DSNGØØ1I Please inpt a valid DB2 subsystem name'
DSNGØØ2  'Line &ZZTDTOP of &ZZTDROWS'    .HELP=*   .ALARM=YES
'
'
DSNGØØ3  'Invalid Type'                  .HELP=*   .ALARM=YES
'DSNGØØ3I Please input a valid type, should either be test or quco'
DSNGØØ5  'Wrong type'                    .HELP=*   .ALARM=YES
'DSNGØØ5I DB2 subsystm name and type didn't match'
DSNGØØ7  'Wrong Lpar'                  .HELP=*   .ALARM=YES
'DSNGØØ7I You must logon to the same lpar as this DB2 subsystem'
DSNGØØ9  'file not found'                .HELP=*   .ALARM=YES
'DSNGØØ9I Please input a valid dataset and member name, don't put '''''
DSNGØ11  'no zparm or decp'              .HELP=*   .ALARM=YES
'DSNGØ11I Check your input file, no zparm or decp found'
```

DSNG01.MSG.MSG

```
DSNGØ11  'no zparm or decp'              .HELP=*   .ALARM=YES
'DSNGØ11I Check your input file, no zparm or decp found'
DSNGØ15  'no differnce found'            .HELP=*   .ALARM=YES
'DSNGØ15I Your source match current db2 settings'
```

GENSRC.REX

```
/* REXX */
MAIN_LINE:
    CALL INITIALIZATION
    CALL PROCESS_ZPARM
    CALL PROCESS_DECP
    CALL TERMINATION
    EXIT ØØ

INITIALIZATION:
ADDRESS ISPEXEC
"VGET DSSN SHARED"
MEMNAM1 = 'DSNZ'||DSSN
MEMNAM2 = 'DSNH'||DSSN
MSG_STATUS = MSG("OFF")
HLQ = SYSVAR(SYSUID)
N = Ø
K = Ø
LM_MACRO_OLD= DSN6SYSP
ADDRESS TSO
"DELETE '"HLQ".DELETE.LISTCAT'"
"FREE FILE(WORK1)"
"ALLOCATE FILE(WORK1) DSN('"HLQ".ZPARM.WORK1') OLD"
"ALLOC F(HOLDIT)  DA('"HLQ".DELETE.LISTCAT') NEW",
   "SPACE(1,1) TRACKS DIR(Ø) UNIT(SYSDA) DSORG(PS) RECFM(V B)",
   "LRECL(125) BLKSIZE(629)"
"LISTC ENTRIES('"HLQ".ZPARM.CNTL') OUTFILE(HOLDIT)"
IF RC > Ø THEN DO
   "ALLOC F(OUTDD)  DA('"HLQ".ZPARM.CNTL') NEW",
      "SPACE(14,1) TRACKS UNIT(SYSDA) CATALOG DIR(1Ø) RECFM(F B)",
      "LRECL(8Ø) DSORG(PO)"
END
"FREE FILE(OUTDD1)"
 "ALLOC FILE(OUTDD1) DSNAME('"HLQ".ZPARM.CNTL("MEMNAM1")') OLD"
"FREE FILE(OUTDD2)"
 "ALLOC FILE(OUTDD2) DSNAME('"HLQ".ZPARM.CNTL("MEMNAM2")') OLD"
"FREE FI(HOLDIT)"
"DELETE '"HLQ".DELETE.LISTCAT'"
RETURN

PROCESS_ZPARM:
CALL FMTWK1
ADDRESS TSO
"EXECIO * DISKR WORK1 (STEM LISTA. FINIS"
DO I=1 TO LISTA.Ø
  IF WORDPOS('DSN6SYSP',LISTA.I)  ¬= Ø THEN LEAVE
END
QUEUE  LEFT('',4)||"DSN6ENV    MVS=XA"
DO J=I TO LISTA.Ø
```

```
   LM_MACRO = SUBWORD(LISTA.J,1,1)
   LM_PARM       = SUBWORD(LISTA.J,2,1)
   LM_VALUE = SUBWORD(LISTA.J,3,1)
   IF LM_PARM = 'WRTHRSH' | LM_PARM ='ALPOOLX' THEN ITERATE
   IF SUBSTR(LISTA.J,34,1) =''  THEN LM_VALUE = ''
   CALL SPECHAND
   IF DATATYPE(LM_VALUE) = 'NUM' THEN LM_VALUE = ABS(LM_VALUE)
      IF LM_MACRO = LM_MACRO_OLD THEN DO
         K=K+1
         WS_PARM.K = LM_PARM
         WS_VALUE.K = LM_VALUE
      END
      ELSE DO
         CALL PRTOUT
         K=1
         WS_PARM.K = LM_PARM
         WS_VALUE.K = LM_VALUE
      END
      IF LM_MACRO = 'DSNHDECP' THEN LEAVE
      LM_MACRO_OLD = LM_MACRO
      N=N+1
END
QUEUE  LEFT('',4)||"END"
COUNT = QUEUED()
"EXECIO "COUNT" DISKW OUTDD1 (FINIS"
RETURN

PROCESS_DECP:
DO P=J TO LISTA.Ø
   LM_MACRO = SUBWORD(LISTA.P,1,1)
   LM_PARM       = SUBWORD(LISTA.P,2,1)
   LM_VALUE = SUBWORD(LISTA.P,3,1)
   IF SUBSTR(LISTA.P,34,1) =''  THEN LM_VALUE = ''
   IF P = J THEN
   LINE_INI = LEFT('',4)||LEFT(DSNHDECM,11)
   ELSE
   LINE_INI = LEFT('',15)
   IF P=LISTA.Ø THEN
   LINE = LINE_INI||LM_PARM||'='||SPACE(LM_VALUE)
   ELSE DO
   LINE = LINE_INI||LM_PARM||'='||SPACE(LM_VALUE)||','
   LINE = LEFT(LINE,71)||'X'
   END
   QUEUE LINE
END
 QUEUE  LEFT('',4)||"END"
    COUNT = QUEUED()
  "EXECIO "COUNT" DISKW OUTDD2 (FINIS"
RETURN
```

```
PRTOUT:
 DO L=1 TO K
 MIN_PARM = WS_PARM.L
 MIN_N=L
    DO M=L+1 TO K
      IF WS_PARM.M = '' THEN ITERATE
      IF WS_PARM.M < MIN_PARM THEN DO
        MIN_PARM = WS_PARM.M
        MIN_N=M
      END
    END
 TEMP_NAME = WS_PARM.L
 WS_PARM.L = MIN_PARM
 WS_PARM.MIN_N = TEMP_NAME
 TEMP_VALUE = WS_VALUE.L
 WS_VALUE.L = WS_VALUE.MIN_N
 WS_VALUE.MIN_N = TEMP_VALUE
 IF L=1 THEN
 LINE_INI = LEFT('',4)||LEFT(LM_MACRO_OLD,11)
 ELSE
 LINE_INI = LEFT('',15)
 IF L=K THEN
 LINE = LINE_INI||MIN_PARM||'='||SPACE(WS_VALUE.L)
 ELSE DO
 LINE = LINE_INI||MIN_PARM||'='||SPACE(WS_VALUE.L)||','
 LINE = LEFT(LINE,71)||'X'
 END
 IF MIN_PARM = 'AAAAA'| MIN_PARM ='AAAAB' THEN            DO
 LINE = LINE_INI||SPACE(WS_VALUE.L)||','
 LINE = LEFT(LINE,71)||'X'
 END
 /* SAY LINE  */
 QUEUE LINE
 END
 RETURN

SPECHAND:
   SELECT
     WHEN LM_PARM = 'SRTPOOL' THEN
          LM_VALUE = LM_VALUE / 1Ø24
     WHEN LM_PARM = 'EDMPOOL' THEN
          LM_VALUE = LM_VALUE / 1Ø24
     WHEN LM_PARM = 'MAXRBLK' THEN
          LM_VALUE = LM_VALUE * 16
     WHEN LM_PARM = 'CHGDC' THEN DO
          LM_VALUE = SUBSTR(LISTA.J,34,32)
          IF LM_VALUE = 1 THEN LM_VALUE = 'NO'
          ELSE LM_VALUE = 'YES'
          END
     WHEN LM_PARM = 'RESTART' THEN DO
```

```
          LM_VALUE = SUBSTR(LISTA.J,34,32)
          LM_PARM = 'AAAAA'
          END
     WHEN LM_PARM = 'ALL-DBNAME' THEN
          LM_PARM = 'AAAAB'
     WHEN LM_PARM = 'SEQCACH' THEN
          IF LM_VALUE = SEQUENTIAL THEN LM_VALUE = SEQ
     WHEN LM_PARM = 'TWOACTV' THEN
          IF LM_VALUE = 2 THEN LM_VALUE = YES
          ELSE LM_VALUE = NO
     WHEN LM_PARM = 'TWOACTV' THEN
          IF LM_VALUE = 2 THEN LM_VALUE = YES
          ELSE LM_VALUE = NO
     WHEN LM_PARM = 'TWOBSDS' THEN
          IF LM_VALUE = 2 THEN LM_VALUE = YES
          ELSE LM_VALUE = NO
     WHEN LM_PARM = 'TWOARCH' THEN
          IF LM_VALUE = 2        THEN LM_VALUE = YES
          ELSE LM_VALUE = NO
     WHEN LM_PARM = 'AUDITST' THEN DO
          LM_VALUE = CVTVALUE(LM_VALUE,LM_PARM)
          IF LM_VALUE = '(1)' THEN LM_VALUE = YES
          IF LM_VALUE = '(Ø)' THEN LM_VALUE = NO
          END
     WHEN LM_PARM = 'MON' THEN DO
          LM_VALUE = CVTVALUE(LM_VALUE,LM_PARM)
          IF LM_VALUE = '(1)' THEN LM_VALUE = YES
          IF LM_VALUE = '(Ø)' THEN LM_VALUE = NO
          END
     WHEN LM_PARM = 'SMFACCT' THEN DO
          LM_VALUE = CVTVALUE(LM_VALUE,LM_PARM)
          IF LM_VALUE = '(1)' THEN LM_VALUE = YES
          IF LM_VALUE = '(Ø)' THEN LM_VALUE = NO
          END
     WHEN LM_PARM = 'SMFSTAT' THEN DO
          LM_VALUE = CVTVALUE(LM_VALUE,LM_PARM)
          IF LM_VALUE = '(1,3,4,5)' THEN LM_VALUE = YES
          IF LM_VALUE = '(Ø)' THEN LM_VALUE = NO
          END
     WHEN LM_PARM = 'TRACSTR' THEN DO
          LM_VALUE = CVTVALUE(LM_VALUE,LM_PARM)
          IF LM_VALUE = '(1,2,3)' THEN LM_VALUE = YES
          IF LM_VALUE = '(Ø)' THEN LM_VALUE = NO
          END
     WHEN LM_PARM = 'ROUTCDE' THEN
          LM_VALUE = CVTVALUE(LM_VALUE,LM_PARM)
     WHEN LM_PARM = 'ARCWRTC' THEN
          LM_VALUE = CVTVALUE(LM_VALUE,LM_PARM)
     WHEN LM_PARM = 'DEALLCT' THEN
          LM_VALUE = CVTVALUE(LM_VALUE,LM_PARM)
```

```
   OTHERWISE  NOP
   END
   RETURN

FMTWK1:
ADDRESS TSO
NEWSTACK
QUEUE "RUN PROGRAM(DSN8ED7) PLAN(DSN8ED7)",
"LIB('DB2.TEST.RUNLIB.LOAD') PARMS('1>DD:WORK1')"
QUEUE "END"
ADDRESS TSO "DSN SYSTEM("DSSN") RETRY(Ø)"
DELSTACK
RETURN

TERMINATION:
"FREE FILE(WORK1)"
"FREE FILE(OUTDD1)"
"FREE FILE(OUTDD2)"
"FREE FILE(OUTDD)"
OUTPUT = HLQ||'.ZPARM.CNTL'
ADDRESS ISPEXEC
"VPUT OUTPUT SHARED"
ADDRESS ISPEXEC
"DISPLAY PANEL(DONE)"
RETURN
```

MAIN.PAN

```
)ATTR
! TYPE(TEXT) INTENS(LOW) SKIP(ON) COLOR(YELLOW)
# TYPE(TEXT) INTENS(HIGH) SKIP(ON) COLOR(TURQ)
~ TYPE(TEXT) INTENS(HIGH) SKIP(ON) color(PINK)
¬ TYPE(INPUT) COLOR(GREEN) HILITE(USCORE)
_ TYPE(INPUT) COLOR(TURQ)
)BODY
#------------------------~ZPARM TOOL MAIN MENU#----------------------
%
+OPTIONS =====>_Z+                                    +Scroll ===> ¬SCRO#
%
%
% 1  +Compare source with current DB2 setting        !Userid . :&ZUSER
%                                                    !Year . . :&ZSTDYEAR
%                                                    !Date . . :&zdATE
% 2  +Regenerate Zparm source                        !Days . . :&ZJDATE
%                                                    !Time . . :&ZTIME
%                                                    !Terminal :&ZTerm
% 3  +Exit                                           !Pf keys  :&ZKEYS
%                                                    !Screen . :&ZSCREEN
%                                                    !Sysid    :&ZSYSID
%                                                         !Appl ID  :&ZAPPLID
%                                                         !Release  :&ZENVIR
%
%
%
#    Licensed Materials  author: Gao Li Jun
#    8888-AØ8 (C) Copyright June,2ØØ3
#
#
)INIT
.ZVARS='(ZCMD)'
&ZPFCTL = OFF
VPUT (ZPFCTL) PROFILE
&SCRO = DATA
)PROC
VER (&ZCMD,NB)
&ZSEL = TRANS (&ZCMD
               1, 'PANEL(compare)'
               2, 'PANEL(REGEN1)'
               3, EXIT
               ' ',' '
               *,'?')
)END
```

POPUP.PAN

```
)ATTR
! TYPE(TEXT) INTENS(LOW) SKIP(ON) COLOR(YELLOW)
# TYPE(TEXT) INTENS(HIGH) SKIP(ON) COLOR(TURQ)
~ TYPE(TEXT) INTENS(HIGH) SKIP(ON) COLOR(PINK)
¬ TYPE(INPUT) COLOR(GREEN) HILITE(USCORE)
* TYPE(INPUT) COLOR(GREEN)
_ TYPE(INPUT) INTENS(LOW) COLOR(TURQ)
@ TYPE(OUTPUT) INTENS(HIGH) COLOR(YELLOW)
)BODY    WINDOW(36,3)
#-----------------------------------
 &EXPL
#-----------------------------------
)INIT
)PROC
VGET EXPL SHARED
)END
```

REGEN1.PAN

```
)PANEL KEYLIST(ISRSPBC,ISR)
)ATTR
! TYPE(TEXT) INTENS(LOW) SKIP(ON) COLOR(YELLOW)
```

```
# TYPE(TEXT) INTENS(HIGH) SKIP(ON) COLOR(TURQ)
~ TYPE(TEXT) INTENS(HIGH) SKIP(ON) COLOR(PINK)
¬ TYPE(INPUT) COLOR(GREEN) HILITE(USCORE)
* TYPE(INPUT) COLOR(GREEN)
_ TYPE(INPUT) INTENS(LOW) COLOR(TURQ)
)BODY
#----------------------------~REGENERATE ZPARM#------------------------
+COMMAND =====>_ZCMD                                     +Scroll ===> ¬SCRO#
%
%
%
!DB2 subsystem name  . . ._DSSN#
+(Specify the DB2 subsystem name like dsn1, dsn2 etc)
%
%
%
%
%
#
)INIT
&SCRO = DATA
&DSSN = ''
)REINIT
&SCRO = DATA
&DSSN = ''
)PROC
VER(&DSSN,NONBLANK,LIST,DSN1,DSN2,DSN3,DSN4,MSG=DSNGØØ7)
IF (&ZSYSID = MVS1)
  VER(&DSSN,NONBLANK,LIST,DSN1,DSN2,MSG=DSNGØØ7)
IF (&ZSYSID= MVS2)
  VER(&DSSN,NONBLANK,LIST,DSN3,DSN4,MSG=DSNGØØB)

VPUT DSSN SHARED
&ZSEL='CMD(GENSRC)'
)END
```

ZPAMCOMP.REX

```
/*    REXX    */
MAIN:
  ADDRESS TSO "TE"
  CALL INITVARS
  CALL FORMAT
  CALL COMPARE
  CALL TERMINATION

INITVARS:
MSG_STATUS = MSG("OFF")
HLQ = SYSVAR(SYSUID)
```

```
NODECP = YES
NOZPARM = YES
RETRY = 1
N=Ø
ADDRESS ISPEXEC
"VGET DSNAM SHARED"
"VGET DTYP SHARED"
"VGET DSSN SHARED"
"VGET ZPOL SHARED"
ADDRESS ISPEXEC
  "TBCREATE TZPMOUT",
  "NAMES(LMMACRO LMPARM LMVALUE SRVALUE) NOWRITE"
   IF DSNAM = '' THEN
      DSNAM = 'DB2.'||DTYP||'.'||DSSN||'.SDSNSAMP.R71Ø(DSNTIJUZ)'
X = SYSDSN("'"DSNAM"'")
   IF X /= 'OK' THEN DO
      ADDRESS ISPEXEC "SETMSG MSG(DSNGØØ9)"
      SIGNAL TERMINATION
   END
ADDRESS TSO
 "FREE FILE(INDD)"
 "ALLOCATE FILE(INDD) DSN('"DSNAM"') SHR"
  "FREE FILE(WORK1)"
  "FREE FILE(WORK2)"
  "FREE FILE(INDD3)"
  "ALLOCATE FILE(WORK1) DSN('"HLQ".ZPARM.WORK1') OLD"
  "ALLOCATE FILE(WORK2) DSN('"HLQ".ZPARM.WORK2') OLD"
 RETURN

FORMAT:
/*
|| ============================================================
|| ROUTINE:   FORMAT            : POPULATE WORK1 AND WORK2
|| ARGUMENTS: DSNAM,DTYP,DSSN   : DB2 SUBSYSTEM NAME, DB2 TYPE,
||                              : ZPARM SOURCE DATASET NAME
|| RETURN:    (NONE)            : NONE REQUIRED
|| ============================================================
*/
ADDRESS TSO
'EXECIO * DISKR INDD (STEM LIST. FINIS'
   DO I=1 TO LIST.Ø
      IF WORDPOS('DSN6ENV',LIST.I)  ¬= Ø THEN DO
      NOZPARM = NO
      LEAVE
      END
   END
   IF NOZPARM = YES THEN DO
     ADDRESS ISPEXEC "SETMSG MSG(DSNGØ11)"
     SIGNAL TERMINATION
   END
   DO J=I TO LIST.Ø
      IF WORDPOS('END',LIST.J)  ¬= Ø THEN LEAVE
      IF INDEX(LIST.J,'DSN6') ¬=Ø THEN
      LIST.J = "               "||SUBWORD(LIST.J,2,2)
      IF INDEX(LIST.J,'RESTART,') ¬=Ø THEN
      LIST.J = "               RESTART="||SUBWORD(LIST.J,1,1)
      IF INDEX(LIST.J,'ALL,') ¬=Ø THEN
      LIST.J = "               ALL-DBNAME="||SUBWORD(LIST.J,1,1)
      QUEUE LIST.J
   END
   IF ZPOL = 'N' THEN DO
      DO K=J TO LIST.Ø
         IF WORDPOS('DSNHDECM',LIST.K)  ¬= Ø THEN DO
         NODECP = NO
         STRGA = "               "||SUBWORD(LIST.K,2,2)
         QUEUE STRGA
         LEAVE
         END
      END
      IF NODECP = YES THEN DO
        ADDRESS ISPEXEC "SETMSG MSG(DSNGØ13)"
        DROPBUF Ø
        SIGNAL TERMINATION
      END
      DO L=K + 1 TO LIST.Ø
        IF WORDPOS('END',LIST.L)  ¬= Ø THEN LEAVE
        QUEUE LIST.L
      END
   END
COUNT = QUEUED()
"EXECIO" COUNT "DISKW WORK2 (FINIS"
ADDRESS TSO
 "FREE FILE(INDD)"
NEWSTACK
QUEUE "RUN PROGRAM(DSN8ED7) PLAN(DSN8ED7)",
"LIB('DB2.TEST.RUNLIB.LOAD') PARMS('1>DD:WORK1')"
QUEUE "END"
ADDRESS TSO "DSN SYSTEM("DSSN") RETRY(Ø)"
DELSTACK
RETURN
COMPARE:
/*
|| ================================================================
|| ROUTINE:   COMPARE          : COMPARE DIFFERENCE BETWEEN SOURCE
||                             : AND CURRENT DB2 PARAMETER VALUE
|| ARGUMENTS: (NONE)           : NONE REQUIRED
|| RETURN:    OUTPUT           : RETURN DIFFERENCE
|| ================================================================
*/
 ADDRESS TSO
```

```
'EXECIO * DISKR WORK1 (STEM LISTA. FINIS'
'EXECIO * DISKR WORK2 (STEM LISTB. FINIS'
DO I=1 TO LISTA.Ø
   IF WORDPOS('DSN6SYSP',LISTA.I)  ¬= Ø THEN LEAVE
END
DO J=I TO LISTA.Ø
   LM_MACRO = SUBWORD(LISTA.J,1,1)
   LM_PARM  = SUBWORD(LISTA.J,2,1)
   LM_VALUE = SUBWORD(LISTA.J,3,1)
   IF ZPOL = Y & LM_MACRO = DSNHDECP THEN DO
   LEAVE
   END
   IF SUBSTR(LISTA.J,34,1) =''  THEN LM_VALUE = ''
   CALL SPECHAND
   IF DATATYPE(LM_VALUE) = 'NUM' THEN LM_VALUE = ABS(LM_VALUE)
   DO K=1 TO LISTB.Ø
      LENGTH1=INDEX(LISTB.K,'=')-1
      START_POS2=INDEX(LISTB.K,'=')+1
      SR_PARM = SUBSTR(LISTB.K,1,LENGTH1)
      IF SR_PARM = LM_PARM THEN DO
         SR_VALUE = GETSRVAL(LISTB.K)
         IF LM_VALUE /= SR_VALUE & LM_VALUE_B /= SR_VALUE THEN DO
            IF LM_PARM = CATALOG & RETRY = 1 THEN DO
            RETRY = Ø
            ITERATE
            END
         LMMACRO = LM_MACRO;LMPARM = LM_PARM
         LMVALUE = LM_VALUE;SRVALUE = SR_VALUE
         ADDRESS ISPEXEC
         "TBADD TZPMOUT"
         END
      LEAVE
      END
      IF K=LISTB.Ø THEN DO
      CALL COMDEFLT LM_PARM,LM_VALUE
      END
    END
END
ADDRESS ISPEXEC
"TBQUERY TZPMOUT ROWNUM(QROWS)"
IF QROWS = Ø THEN "SETMSG MSG(DSNGØ15)"
"TBTOP TZPMOUT"
 DO UNTIL ENDKEY = 'YES'
   "TBDISPL TZPMOUT PANEL(COMPOUT)"
    IF RC = 8 THEN ENDKEY = "YES"
    "VGET LCMD SHARED"
    LCMD = TRANSLATE(LCMD)
    IF LCMD= S THEN DO
        "VGET LMMACRO SHARED"
        "VGET LMPARM SHARED"
        DO J=1 TO LISTA.Ø
           LM_MACRO = SUBWORD(LISTA.J,1,1)
           LM_PARM  = SUBWORD(LISTA.J,2,1)
           LM_EXPL = STRIP(SUBSTR(LISTA.J,74,36))
           IF LM_MACRO = LMMACRO & LM_PARM= LMPARM THEN DO
              EXPL = LM_EXPL
              "VPUT EXPL SHARED"
              LEAVE
           END
        END
        DO UNTIL ENDKEY1 = 'YES'
           "ADDPOP"
           ZWINTTL = 'EXPLANATION OF THIS PARM'
           "DISPLAY PANEL(POPUP)"
           IF RC = 8 THEN ENDKEY1 = "YES"
           "REMPOP"
        END
   END
   END
RETURN


SPECHAND:
   LM_VALUE_B = ''
   SELECT
     WHEN LM_PARM = 'SRTPOOL' THEN
          LM_VALUE = LM_VALUE / 1Ø24
     WHEN LM_PARM = 'EDMDSPAC' THEN
          LM_VALUE = LM_VALUE / 1Ø24
     WHEN LM_PARM = 'EDMDSMAX' THEN
          LM_VALUE = LM_VALUE / 1Ø24
     WHEN LM_PARM = 'EDMPOOL' THEN
          LM_VALUE = LM_VALUE / 1Ø24
     WHEN LM_PARM = 'MAXRBLK' THEN
          LM_VALUE = LM_VALUE * 16
     WHEN LM_PARM = 'CHGDC' THEN DO
          LM_VALUE = SUBSTR(LISTA.J,34,32)
          IF LM_VALUE = 1 THEN LM_VALUE = 'NO'
          ELSE LM_VALUE = 'YES'
          END
     WHEN LM_PARM = 'RESTART' THEN
          LM_VALUE = SUBSTR(LISTA.J,34,32)
     WHEN LM_PARM = 'SEQCACH' THEN
          IF LM_VALUE = SEQUENTIAL THEN LM_VALUE = SEQ
     WHEN LM_PARM = 'TWOACTV' THEN
          IF LM_VALUE = 2 THEN LM_VALUE = YES
          ELSE LM_VALUE = NO
     WHEN LM_PARM = 'TWOARCH' THEN
          IF LM_VALUE = 2        THEN LM_VALUE = YES
          ELSE LM_VALUE = NO
     WHEN LM_PARM = 'AUDITST' THEN DO
          LM_VALUE = CVTVALUE(LM_VALUE,LM_PARM)
          IF LM_VALUE = '(1)' THEN LM_VALUE_B = YES
          IF LM_VALUE = '(Ø)' THEN LM_VALUE_B = NO
          END
     WHEN LM_PARM = 'MON' THEN DO
          LM_VALUE = CVTVALUE(LM_VALUE,LM_PARM)
          IF LM_VALUE = '(1)' THEN LM_VALUE_B = YES
          IF LM_VALUE = '(Ø)' THEN LM_VALUE_B = NO
          END
     WHEN LM_PARM = 'SMFACCT' THEN DO
          LM_VALUE = CVTVALUE(LM_VALUE,LM_PARM)
          IF LM_VALUE = '(1)' THEN LM_VALUE_B = YES
          IF LM_VALUE = '(Ø)' THEN LM_VALUE_B = NO
          END
     WHEN LM_PARM = 'SMFSTAT' THEN DO
          LM_VALUE = CVTVALUE(LM_VALUE,LM_PARM)
          IF LM_VALUE = '(1,3,4,5)' THEN LM_VALUE_B = YES
          IF LM_VALUE = '(Ø)' THEN LM_VALUE_B = NO
          END
     WHEN LM_PARM = 'TRACSTR' THEN DO
          LM_VALUE = CVTVALUE(LM_VALUE,LM_PARM)
          IF LM_VALUE = '(1,2,3)' THEN LM_VALUE_B = YES
          IF LM_VALUE = '(Ø)' THEN LM_VALUE_B = NO
          END
     WHEN LM_PARM = 'ROUTCDE' THEN
          LM_VALUE = CVTVALUE(LM_VALUE,LM_PARM)
     WHEN LM_PARM = 'ARCWRTC' THEN
          LM_VALUE = CVTVALUE(LM_VALUE,LM_PARM)
     WHEN LM_PARM = 'DEALLCT' THEN
          LM_VALUE = CVTVALUE(LM_VALUE,LM_PARM)
   OTHERWISE  NOP
   END
   RETURN


GETSRVAL:
/*
|| ==============================================================
|| ROUTINE:   GETSRVAL         : EXTRACT SOURCE VALUE FROM WORK2
|| ARGUMENTS: SR_VALUE         : SOURCE VALUE
|| RETURN: OUTPUT_SR_VALUE     : PROCESSED SOURCE VALUE
|| ==============================================================
*/
  ARG INPUT_SR_VALUE
  A =WORDINDEX(INPUT_SR_VALUE,1)
  B =INDEX(INPUT_SR_VALUE,'=')
  C =WORDLENGTH(INPUT_SR_VALUE,1)
  LENGTH2 = A + C - B - 1
  START_POS2=INDEX(INPUT_SR_VALUE,'=')+1
  LAST_CHAR_POS= A + C -1
  LAST_CHAR = SUBSTR(INPUT_SR_VALUE,LAST_CHAR_POS,1)
  LENGTH3 = LAST_CHAR_POS - START_POS2
  IF LAST_CHAR = ',' THEN
     OUTPUT_SR_VALUE = SUBSTR(INPUT_SR_VALUE,START_POS2,LENGTH3)
  ELSE
     OUTPUT_SR_VALUE = SUBSTR(INPUT_SR_VALUE,START_POS2,LENGTH2)
  IF DATATYPE(OUTPUT_SR_VALUE) = 'NUM' THEN
     OUTPUT_SR_VALUE = ABS(OUTPUT_SR_VALUE)
  RETURN OUTPUT_SR_VALUE


COMDEFLT:
/*
|| ===============================================================
|| ROUTINE:   COMDEFLT         : COMPARE WITH DEFAULT TABLE IF NOT
||                             : FOUND IN ZPARM SOURCE
|| ARGUMENTS: LM_PARM LM_VALUE : PARAMETER NAME AND ITS CURRUENT
||                             : VALUE
|| RETURN: NONE                : NONE REQUIRED
|| ===============================================================
*/
ARG WS_PARM_NAME,WS_PARM_VALUE
ADDRESS TSO
"FREE FILE(INDD3)"
"ALLOCATE FILE(INDD3) DSN('SYSLXG.ZPARM.EXEC(DEFAULT)') SHR"
ADDRESS TSO 'EXECIO * DISKR INDD3 (STEM LISTC. FINIS'
DO L=1 TO LISTC.Ø
   LENGTH=INDEX(LISTC.L,'=')-1
   START_POS2=INDEX(LISTC.L,'=')+1
   DF_PARA_NAME = SUBSTR(LISTC.L,1,LENGTH)
   IF DF_PARA_NAME = WS_PARM_NAME THEN DO
      DF_CURR_VALUE = GETSRVAL(LISTC.L)
      IF WS_PARM_VALUE /= DF_CURR_VALUE THEN DO
         LMMACRO = LM_MACRO
         LMPARM  = LM_PARM
         LMVALUE = LM_VALUE
         SRVALUE = DF_CURR_VALUE
         ADDRESS ISPEXEC
         "TBADD TZPMOUT"
      END
      LEAVE
   END
   IF L=LISTC.Ø THEN SAY  WS_PARM_NAME "IS A NEW PARM, UPDATE DEFAULT"
END
ADDRESS TSO "FREE FILE(INDD3)"
RETURN

TERMINATION:
 ADDRESS TSO
```

```
"FREE FILE(WORK1)"
"FREE FILE(WORK2)"
"FREE FILE(INDD3)"
"FREE FILE(INDD)"
 ADDRESS ISPEXEC "TBCLOSE TZPMOUT"
 EXIT
```

COMPARE.PAN

```
)PANEL KEYLIST(ISRSPBC,ISR)
)ATTR
! TYPE(TEXT) INTENS(LOW) SKIP(ON) COLOR(YELLOW)
# TYPE(TEXT) INTENS(HIGH) SKIP(ON) COLOR(TURQ)
~ TYPE(TEXT) INTENS(HIGH) SKIP(ON) COLOR(PINK)
¬ TYPE(INPUT) COLOR(GREEN) HILITE(USCORE)
* TYPE(INPUT) COLOR(GREEN)
_ TYPE(INPUT) INTENS(LOW) COLOR(TURQ)
)BODY
#----------------------------~COMPARE PANEL#--------------------------
+COMMAND =====>*ZCMD                                 +Scroll ===> ¬SCRO#
%
%
%
%
!DB2 subsystem name  . . ._DSSN#
+(Specify the DB2 subsystem name like dsn3, dsn7 etc)
%
%
%
%
%
%
!DB2 Subsystem type  . . ._DTYP#
+(Specify the DB2 subsystem type like test, quco or prod)
%
%
#
)INIT
&SCRO = DATA
&dssn = ''
&dtyp = ''
)REINIT
&SCRO = DATA
&DSSN = ''
&DTYP = ''
)PROC
VER(&DTYP,NONBLANK,LIST,TEST,QUCO,MSG=DSNGØØ3)
IF (&DTYP = TEST)
  VER(&DSSN,NONBLANK,LIST,DSN1,DSN2,MSG=DSNGØØ5)
```

```
IF (&DTYP = QUCO)
  VER(&DSSN,NONBLANK,LIST,DSN3,DSN4,MSG=DSNGØØ5)
IF (&ZSYSID = MVS1)
  VER(&DSSN,NONBLANK,LIST,DSN1,DSN2,MSG=DSNGØØ7)
IF (&ZSYSID= MVS2)
  VER(&DSSN,NONBLANK,LIST,DSN3,DSN4,MSG=DSNGØØ7)

VPUT DSSN SHARED
VPUT DTYP SHARED
&ZSEL='PANEL(COMPARE1)'
)END
```

Lijun Gao (legend_gao@yahoo.com)
Senior DB2 System Programmer (USA)

DB2 UDB V8 LUW – the SET operators

This article looks at the SET operators that are available in DB2 UDB for LUW. I ran the following SQL on a Windows 2000 machine running DB2 V8 FP2.

Three SET operators are available – UNION, EXCEPT, and INTERSECT. I will look at each of these in turn.

For all of our examples we will use two tables, hm01 and hm02, created and populated as follows:

```
create table hmØ1 (id int, name char(1Ø))
create table hmØ2 (id int, name char(1Ø))

insert into hmØ1 values(1,'AAA'),(2,'BBB'),(3,'CCC')
insert into hmØ2 values(1,'AAA'),(4,'DDD'),(5,'EEE')
```

The UNION operator lets you join the results of two SELECT statements, with duplicate rows being eliminated (hence requiring a sort). The UNION ALL operator does not eliminate duplicate rows (and hence does not require a sort).

So if we use the UNION operator and select everything from our two tables, you can see that the operator returns only unique rows (one of the duplicate ‘1 AAA’ rows has been eliminated) and the number of rows returned is 5:

```
>db2 select * from hmØ1 union select * from hmØ2

ID          NAME
---------- ----------
         1 AAA
         2 BBB
         3 CCC
         4 DDD
         5 EEE

  5 record(s) selected.
```

If we now use the UNION ALL operator, we can see we have two occurrences of the ‘1 AAA’ row and the number of rows returned is 6:

```
>db2 select * from hmØ1 union all select * from hmØ2

ID          NAME
---------- ----------
         1 AAA
         4 DDD
         5 EEE
         1 AAA
         2 BBB
         3 CCC

  6 record(s) selected.
```

If you do a visual explain on both of the above statements using the Control Centre, you can see that the UNION version does a SORT whereas the UNION ALL version doesn’t.

Now let’s look at the EXCEPT operator. The EXCEPT operator returns rows in the table to the left of the operator which do not exist in the table to the right of the operator. The operator is used as shown in the example below:

```
>db2 select * from hmØ1 except select * from hmØ2

ID          NAME
---------- ----------
         2 BBB
         3 CCC
```

You can see that the result returned is the two rows in table hm01 that are not present in table hm02.

Finally, let’s look at the INTERSECT operator. As the name suggests, and in line with normal mathematical set theory, the INTERSECT operator will return only rows that exist in both tables specified either side of the operator. This is shown in the example below:

```
>db2 select * from hmØ1 intersect select * from hmØ2

ID         NAME
---------- ----------
         1 AAA

  1 record(s) selected.
```

The only row that is present in both table hm01 and hm02 is the ‘1 AAA’ row, and this is the row that is returned.

You can use the additional ALL parameter for both the EXCEPT and INTERSECT operators. The manual says that this works in the same way as with UNION and UNION ALL. However, there is no difference as far as I can see between EXCEPT and EXCEPT ALL and INTERSECT and INTERSECT ALL. If you specified EXCEPT ALL in the third example, you would still get only the two rows back, and if you specified INTERSECT ALL in the fourth example, you would still get the only one row returned.

I hope I have shown how to use the three set operators UNION, EXCEPT, and INTERSECT.

C Leonard
Freelance Consultant (UK)

DB2 news

Princeton Softech has announced Archive for DB2 PeopleSoft Edition. PeopleSoft DB2 mainframe sites can now use its database archiving capabilities and predefined archive templates including business objects such as vouchers, ledgers, and payroll. Archive templates separate current from historical transactions and make it easy to implement database archiving without incurring the expense and risk of additional technical design and development.

By targeting the most data intensive PeopleSoft applications for archiving, including related tables to maintain the referential integrity of the data, Archive for DB2 PeopleSoft Edition enables companies to archive rarely accessed historical data and remove this data from the DB2 production database. Archiving increases application availability and performance while giving companies selective and on-demand access to archived data that can be stored on the most convenient and cost effective storage medium.

For further information contact:
Princeton Softech, 111 Campus Drive, Princeton, NJ 08540-6400, USA.
Tel: (609) 627 5500.
URL: http://www.princetonsoftech.com/news/press/a4db2-peoplesoft.htm.

* * *

HiT Software has announced Version 2.0 of Ritmo, which it describes as a "100% managed .NET data provider for IBM DB2 access". The product allows developers to access DB2 databases within the Microsoft .NET framework in order to deliver applications with maximum performance, scalability, reliability, and security.

Ritmo allows developers to submit SQL statements that more efficiently access, retrieve, and update DB2 data. Additionally, it delivers connection pooling, bulk data insert, and data compression to maximize performance. New data type support for BLOB/CLOB and BIGINT, as well as double-byte character set support, let applications target a larger number of users.

For further information contact:
HiT Software, 4020 Moorpark Avenue, Suite 100, San Jose, CA 95117, USA.
Tel: (408) 345 4001.
URL: http://www.hitsw.com/products_services/sqldb2/ritmodb2/ritmodb2.html.

* * *

IBM has released a software patch for a security vulnerability in two components of DB2 Version 7.2 for Linux. Without the patch, attackers could run malicious code on DB2 systems using the permissions of an administrative (root) account.

The buffer-overflow vulnerabilities aren't accessible to remote users. Attackers first have to connect to DB2 on a corporate intranet with a user account to launch an attack.

For further information contact your local IBM representative.