145

DB2

update

November 2004

In this issue

A xephon PUBLICATION

DB2 Update

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

Bob Thomas
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 2000 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.

This issue is dedicated to the memory of Chris Bunyan, co-founder of Xephon and creator of the *Update* journals.

Printed in England.

Some useful features available in DB2 LOAD

DB2 V7 has introduced two new and very useful features that can be used by application programmers in populating their DB2 tables. The features are LOAD with SHRLEVEL CHANGE and LOAD using dynamic SQL.

SHRLEVEL CHANGE

With the SHRLEVEL CHANGE option, applications can concurrently read and write to a tablespace or partition during LOAD. The default option is still SHRLEVEL NONE, which does not allow concurrent access to application programs. The SHRLEVEL CHANGE option works only with the RESUME YES option. The syntax is:

```
LOAD DATA
INDDN SYSREC
RESUME YES
SHRLEVEL CHANGE
INTO TABLE C1.T1
```

This option does not allow the specification of the parameters INCURSOR, RESUME NO, REPLACE, KEEPDICTIONARY, LOG NO, ENFORCE NO, SORTKEYS, STATISTICS, COPYDDN, RECOVERYDDN, PREFORMAT, REUSE, and PART integer REPLACE.

With this option, DB2 does not perform SORT, BUILD, SORTBLD, INDEXVAL, ENFORCE, or REPORT phases.

LOAD with SHRLEVEL CHANGE really functions like a mass insert job. It tries to insert the new records into available free space as close to the clustering index as possible. If insert triggers are defined on the table, it fires those triggers. It generates log records for the purpose of recoverability and does not leave the tablespace/partition in COPY-pending restricted mode. If a very large number of records are loaded, this may require a REORG to improve the clustering of the tablespace.

When LOAD starts, it puts the tablespace or partition, and index or index partition, in RW, UTRW mode and makes a write claim on the object. If for any reason the job fails, it still allows application programs to continue accessing the table in read/write mode. Once the reason for the failure is identified and corrected, the job can be restarted and LOAD resumes loading data from the restart point.

This option is useful for using LOAD as a general purpose application program – like any application program developed with COMMITs and RESTARTABILITY features. It is used for populating DB2 tables without the need to develop such a program. This option also allows all input records discarded during loading (because of a variety of errors) to be saved.

LOAD USING DYNAMIC SQL

With the dynamic SQL option, the LOAD utility allows input records to be read directly from another table (which has the same/compatible structure as the table being loaded), which may belong to the same or a different subsystem. This is achieved by declaring a cursor on the input DB2 table using an EXEC SQL statement and then specifying the INCURSOR option of the LOAD utility in the control statement.

One example is:

```
//SYSIN    DD *
  EXEC SQL
           DELETE     FROM C1.T1
           WHERE      T1.COL1 = 'US'
  ENDEXEC

  EXEC SQL
           DECLARE CUR1 CURSOR FOR
               SELECT * FROM LOC1.C2.T2
               WHERE      T2.COL1 = 'US'
  ENDEXEC

  LOAD DATA
            LOG YES
            RESUME YES
            INCURSOR (CUR1)
            INTO TABLE
```

```
                C1.T1
//
```

In this example, the job first deletes all rows having a value of ‘US’ in column COL1 in table T1. Then it declares a cursor, CUR1, to retrieve all rows having a value of ‘US’ for column COL1 in table T2 residing in a different location, LOC1. Finally, with LOAD statements, it loads the rows in table T1 using the cursor CUR1 declared in the previous step.

This option does not allow loading into the same table as where the cursor is defined. The INCURSOR option is incompatible with the SHRLEVEL CHANGE option. At execution time, the SQL statement string is parsed and checked for any errors. For invalid SQL statements, the error condition that prevented the execution is reported.

Another example of INCURSOR using partition parallelism is:

```
//SYSIN   DD *
EXEC SQL
         DECLARE C00001 CURSOR FOR
             SELECT  *  FROM  C1.T1
                  WHERE TBL_PTN_NR = 1
ENDEXEC

EXEC SQL
         DECLARE C00003 CURSOR FOR
             SELECT  *  FROM  C1.T1
                  WHERE TBL_PTN_NR = 3
ENDEXEC

EXEC SQL
         DECLARE C00004 CURSOR FOR
             SELECT  *  FROM  C1.T1
                  WHERE TBL_PTN_NR = 4
ENDEXEC

LOAD DATA
     LOG YES
     INTO TABLE  C2.T2   PART 1 REPLACE   INCURSOR (C00001)
     INTO TABLE  C2.T2   PART 3 REPLACE   INCURSOR (C00003)
     INTO TABLE  C2.T2   PART 4 REPLACE   INCURSOR (C00004)
//
```

This option is useful for populating DB2 tables without requiring any UNLOAD data from the source table and then saving the

unloaded data in a dataset. This is a very practical approach for reading input rows from a table that exists in another DB2 subsystem and loading these rows into the target table. Compared with the INSERT INTO ….. SELECT FROM ….. SQL command, this option provides better data organization and other benefits of using the DB2 LOAD utility.

Sharad Pande
Senior DBA (USA)

SMF exit for DB2

Alongside the normal DB2 applications that run against our production DB2 datasharing environment we have an ever-growing workload connecting via TCP/IP and DDF. The problem we have is that for our traditional applications we wish to collect SMF accounting records, but for the distributed workload we do not. The existing tracing controls within DB2 offer the facility to start a trace for a named plan or, indeed, a list of plans (up to a limit), but not the facility to exclude a named plan from being traced. The sheer volume of accounting records produced by our distributed workload has, on occasion, flooded the SMF buffers with the resulting loss of data not only for DB2 but for other software that writes to SMF. Along with this, the SMF housekeeping and reporting processes are effectively working on large volumes of data – a significant proportion of which is not required.

As mentioned, the ideal solution would be an EXCLUDE option within the DB2 TRACE command because this would stop the records being written at source. The added advantage of this would be saving on CPU overhead caused by collecting and writing the trace record in the first place. At present this is not an option, so the next best thing is to intercept the accounting records before SMF collects them. The following code is an

SMF exit that looks for specific PLAN names within DB2 101 accounting records and directs SMF to ignore them. The source code contains a table of PLAN names that we want SMF to ignore. For our distributed applications, the PLAN name is a constant, DISTSERV, and is the only one to feature in our list. Sample JCL is supplied giving details of installing this exit as a z/OS user mod.

The software levels involved are z/OS 1.4 and DB2 V6 and V7, although the code should be valid for most combinations of MVS and DB2.

SOURCE CODE FOR THE SMF EXIT

```
IEFU83 TITLE 'SMF RECORD EXIT'
*********************************************************************
*   DESCRIPTION
*   THIS ROUTINE IS USED FOR IEFU83/4/5 TO SUPPRESS DB2 RECORD
*   TYPE 1Ø1S FOR SELECTED PLAN NAMES.
*   THERE IS A HARDCODED TABLE OF PLAN NAMES IN THIS CSECT.
*   INTERFACE
*   ENTRY POINT = MODU83
*   ENTRY REGISTERS = REG1 POINTS TO A 4-BYTE ADDRESS OF THE
*                     RECORD DESCRIPTOR WORD (RDW) OF THE SMF
*                     RECORD TO BE PROCESSED BY THE EXIT.
*   RETURN CODES = Ø - WRITE THE SMF RECORD (KEEP)
*                  4 - DO NOT WRITE THE SMF RECORD (DELETE)
*   ATTRIBUTES
*          LOCATION = LPA
*          STATE = SUPERVISOR
*          AMODE = 31
*          RMODE = ANY
*          KEY = KEY Ø
*          MODE = ANY (CAN BE SRB OR XMEM, LOCKED)
*          SERIALIZATION = NONE
*          TYPE = REENTRANT
*          NAME = IEFU83
*          ALIAS = IEFU84, IEFU85
*  !!! WARNING !!!
*  BECAUSE THE EXIT MAY BE INVOKED IN SRB MODE (IEFU84/5) IT CANNOT
*  ISSUE ANY SVCS.
*********************************************************************
         EJECT
MODU83   CSECT
MODU83   AMODE 31
MODU83   RMODE ANY
```

```
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12                  BASE REG
R13      EQU   13                  SAVEAREA REG
R14      EQU   14                  RETURN ADDRESS
R15      EQU   15                  ENTRY ADDRESS & TEMPORARY BASE
         USING MODU83,R15
         SAVE  (14,12),,'MODU83 TESCO &SYSDATE &SYSTIME'
         LR    R12,R15             LOAD BASE ADDR
         DROP  R15
         USING MODU83,R12
         LR    R9,R1               SAVE PARAMETER LIST ADDRESS
         L     R9,0(R9)            GET ADDRESS OF SMF RECORD
         USING SM101,R9
         SLR   R15,R15             SET RC = ACCEPT RECORD
         CLI   SM101RTY,101        IS IT TYPE 101
         BNE   RETURN              NO, LET IT WRITE
*********************************************************************
*  SUPPRESS DB2 TYPE 101 FOR PLAN 'DISTSERV'
*********************************************************************
         LA    R3,SM101END         TO START OF SELF-DEFINING SECTION
         USING QWA0,R3
         LA    R4,0(R9)            START OF SMF RECORD (RESET BIT0)
         A     R4,QWA01PSO         TO PRODUCT SECTION
         LH    R0,QWA01PSL         LEN OF PRODUCT SECTION
         LR    R3,R4               START OF PRODUCT SECTION
         USING QWHS,R3
         AR    R4,R0               END OF PRODUCT SECTION
NXTHDR   DS    0H
         CLI   QWHSTYP,QWHSHC02    CORRELATION HEADER ?
         BE    CHKPLAN             YES, CHECK PLAN NAME
         AH    R3,QWHSLEN          TO NEXT HEADER
         CR    R3,R4               ANY MORE HEADERS
         BL    NXTHDR              YES, KEEP LOOKING
         B     RETURN              DO NOT SUPPRESS
*
* CHECK PLAN NAME AGAINST TABLE OF THOSE WE SUPPRESS
*
         USING QWHC,R3
CHKPLAN  DS    0H
```

```
         LA    R1,PLANTAB
CHKNXT   CLI   Ø(R1),C' '          END OF TABLE ?
         BE    RETURN              YES, KEEP SMF RECORD
         CLC   QWHCPLAN,Ø(R1)      PLAN NAME MATCH ?
         LA    R1,L'QWHCPLAN(R1)   TO NEXT ENTRY
         BNE   CHKNXT              NO, KEEP LOOKING
         LA    R15,4               SET RC = SUPPRESS RECORD
*
*  RETURN WITH RC IN R15
*
RETURN   DS    ØH
         L     R14,12(R13)         RESTORE RETURN ADDRESS
         LM    RØ,R12,2Ø(R13)      RESTORE REGS ZERO THRU 12
         BSM   Ø,R14               RESTORE CALLER MODE AND RETURN
*
PLANTAB  DS    ØH
         DC    CL8'DISTSERV'
         DC    CL8' '              SPARES
         DC    CL8' '
         DC    CL8' '
         DC    CL8' '
         DC    C' '                END-OF-TABLE
         DSNDQWAS SUBTYPE=ALL
         END
```

JOB TO INSTALL SMF EXIT

```
//JOBCARD
//S1       EXEC PGM=GIMSMP,
//         PARM='PROCESS=WAIT',
//         DYNAMNBR=12Ø
//SMPCSI   DD DISP=SHR,DSN=SMPE.ZOS14.GLOBAL.CSI
//SYSLIB   DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=SYS1.MODGEN,DISP=SHR
//         DD DSN=DB2.V71SMP.SDSNMACS,DISP=SHR
//SMPHOLD  DD DUMMY
//SMPCNTL  DD DDNAME=SYSIN
//SYSIN    DD  *
  SET BDY(GLOBAL).
  REJECT  S(TMSMFØ7) BYPASS(APPLYCHECK) .
  RESETRC.
  RECEIVE S(TMSMFØ7) .
  SET BDY(TGT14A) .
  APPLY S(TMSMFØ7) REDO CHECK RC(RECEIVE=Ø).
  APPLY S(TMSMFØ7) REDO RC(APPLY=Ø).
//SMPPTFIN DD  DATA,DLM=ŁŁ
++USERMOD(TMSMFØ7) REWORK(2ØØ41381) /* IDENTIFY CHANGE DATE YYYYDDDV */.
++VER(ZØ38) FMID(HBB77Ø7)
   /*****************************************************************
```

```
  *
  *:ELEMENTS:     SRC - MODU83
  *:DESCRIPTION:  SMF EXIT
  *:FUNCTION:     SMFWTM (SMF RECORD WRITE) EXIT TO SUPPRESS
  *               SELECTED DB2 (TYPE 101) RECORDS.
  *
   ************************************************************/.
++JCLIN.
//LINK       EXEC PGM=IEWL,PARM='LET,NCAL,REUS,RENT,AC=1'
//SYSLMOD    DD   DSN=SYS1.LPALIB,DISP=SHR
//SYSLIN     DD   *
 INCLUDE SYSPUNCH(MODU83)
 MODE AMODE(31),RMODE(ANY)
 ENTRY  MODU83
 NAME   MODU83(R)
/*
++SRC(MODU83) DISTLIB(TESSRCE) TXLIB(TESSRCE) .
ff
```

Steve Kemp (UK)

Application contention report from DB2 for z/OS

It is sometimes useful to collect the DEADLOCK and TIMEOUT messages from the DB2 MSTR address space and collate them to summarize which transactions are giving trouble. DB2CONT is a REXX program to do just that, written in conjunction with James Gill from Triton Consulting.

Details are taken from multiple messages, then combined, reformatted, and summarized to make them easier to read. DB2CONT also shows any lock escalation or dataset extend failure messages in its report, and that could be easily extended to collate other messages too.

SAMPLE REPORT

Here is part of a sample report produced by DB2CONT, to show how it looks:


```
REPORT FOR MONDAY, Ø7 JUN 2ØØ4
CONTENTION REPORT - OBJECTS
Object                          DLocks TOuts
======                          ====== =====
DCSØØ1P1.SFINPLC                   23   133
DCSØØ1P1.SFINPLP                    1    12
DCSØØ1P1.SFINFSS                    Ø     2
CONTENTION REPORT - TRANSACTIONS/JOBS
Victim    DLocks    TOuts
======    ======    =====
F417           4       25
F42Ø           2       12
F421           3        8
Holder    DLocks    TOuts
======    ======    =====
E123          23      111
F42Ø           1        Ø
NØØ8           Ø        1
Victim    Holder    DLocks    TOuts
======    ======    ======    =====
F417      E123           4       25
F42Ø      E123           2       11
PFF1Ø946  PFF1Ø945       1        Ø
Deadlocks
==========
Ø8.39.Ø1 Plan PCSP1A Trans E123 in PA31CICS with Plan PCSP1A Trans F42Ø
          in PA41CICS on (ØØØØ21Ø) DCSØØ1P1.SFINPLC .ØØØØØØØ1
Ø9.51.32 Plan PCSP1A Trans E123 in PA32CICS with Plan PCSP1A Trans E123
          in PA31CICS on Tablespace DCSØØ1P1.SFINPLC page X'Ø5EØ1ØA7
Ø9.52.25 Plan PCSP1A Trans E134 in PA32CICS with Plan PCSP1A Trans E124
          in PA41CICS on Tablespace DCSØØ1P1.SFINCSH page X'ØØØ172
Timeouts
========
Ø8.13.41 Plan PCSP1A Trans F417 in PA31CICS to Plan PCSP1A Trans E123
          in PA12CICS on Tablespace DCSØØ1P1.SFINPLC page X'Ø66Ø2766
Ø8.13.51 Plan PCSP1A Trans F42Ø in PA32CICS to Plan PCSP1A Trans E123
          in PA12CICS on Tablespace DCSØØ1P1.SFINPLC page X'Ø67Ø1A2C
11.1Ø.32 Plan PPFØØ1F1 Job PFF1Ø946 in BATCH with Plan PPFØØ1F1 Job
PFF1Ø945
          in BATCH on Tablespace DPFØØ1F1.SCSEPER page X'5ØØØØ3'
Trace data was missed by the online monitor 6 time(s).
REPORT FOR TUESDAY, Ø8 JUN 2ØØ4
CONTENTION REPORT - OBJECTS
Object                          DLocks TOuts
======                          ====== =====
DCSØØ1P1.SFINFSH                   31   171
   ..
   ..
   ..
   ..
```

```
REPORT FOR SATURDAY, 12 JUN 2004
Trace data was missed by the online monitor 4 time(s).
Lock escalation occurred 31 time(s):
  Time    Object locked    Lock  Connectn  Correlatn   Plan      Package
========  ============    ====  ========  =========   ====      =======
01.00.36  OCS001P1.CDTACC  'X'   DB2CALL   PDB0CAPT   ASNLP710  ASNLDM71
02.42.25  ASN.IBMSNAP_UOW  'X'   DB2CALL   PDB0CAPT   ASNLP710  ASNLDM71
02.49.16  OCS001P1.TINP_D  'X'   RRSAF     CSP1093A   PCSP1A    IBTD0401
Datasets failed to extend 11 time(s):
Dataset                                                 Count
=======                                                 =====
PDB0.DSNDBD.WORKPDB3.DSN4K03.I0001.A002.                    1
PDB0.DSNDBD.WORKPDB3.DSN4K10.I0001.A002.                    1
CONTENTION REPORT - SUMMARY
Victim    Holder    DLocks    TOuts
======    ======    ======    =====
F417      E123          16      126
F420      E123           6       45
PFF10946  PFF10945       1        0
                         Dlocks    TOuts
                         ======    =====
MONDAY,    07 JUN 2004      32      186
TUESDAY,   08 JUN 2004     120      706
WEDNESDAY, 09 JUN 2004      54      385
THURSDAY,  10 JUN 2004      63      358
FRIDAY,    11 JUN 2004      53      494
SATURDAY,  12 JUN 2004       0        0
```

The DB2 MSTR task usually runs for more than one day and the figures are collated for each day separately.

The date is found in the input files at the start each day's messages.

The report shows the full correlation-id (ie jobname) for batch jobs and the 4-byte name for CICS transactions (part of the correlation- id).

At the end of the report there is a CONTENTION REPORT - SUMMARY, which summarizes all the deadlocks and timeouts for all the days.

SAMPLE JOB

Here is a sample job to extract the output from the 'dsn1mstr' task via batch SDSF, followed by a run of DB2CONT to produce a report:

```
//jobname JOB  (),'REPORT ON DB2 (dsn1)',CLASS=D,MSGCLASS=X,REGION=4M
//*
//SDSF  EXEC PGM=SDSF,PARM='++24,8Ø'
//ISFOUT  DD SYSOUT=*
//ISFIN   DD *
DA ALL
PREFIX *
FIND dsn1mstr
++?
FIND JESMSGLG
++S
PRT ODSN 'sdsf.output.dataset.name' * OLD
PRT
PRT CLOSE
/*
//REPORT EXEC PGM=IKJEFTØ1,REGION=8M,
//            COND=(Ø,LT)
//SYSPROC  DD DISP=SHR,DSN=your.exec.library
//JLØ1     DD DISP=SHR,DSN=sdsf.output.dataset.name
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
%DB2CONT
/*
```

If you are running this for a datasharing group, you can run the job once for each subsystem. We cut and pasted some parts of the report into a spreadsheet to combine the figures for all the members of the group, but that won't be described in this article.

DB2CONT PROGRAM

```
/* Rexx ---------------------------------------------------------------*/
/*                                                                     */
/* DB2CONT                                                             */
/* =======                                                             */
/* This EXEC is intended to post-process the DB2 STC job log. It       */
/* is run against a dataset populated by a batch run of SDSF.          */
/*                                                                     */
/* There are no input parameters.                                      */
/*                                                                     */
/* The EXEC reads through the job log collating messages to produce    */
/* a user-friendly report.                                             */
/*                                                                     */
/* See the end of this member for a list of the messages supported.    */
/*                                                                     */
/* Rexx ---------------------------------------------------------------*/
   call initialise
   call get_joblog
```

```
   call process_joblog
   call print_report
   call report_summary
   /* call report_others */
Exit max_cc
/* initialise ----------------------------------------------------*/
/* This procedure is used to initialize all the code storage.     */
/* initialise ----------------------------------------------------*/
initialise:
   signal on error
   newstack             /* start a new (empty) stack              */
   true = 1
   false = 0
   not_procd. = ""      /* DB2 Message ids not processed          */
   not_procd.0 = 0      /* Count of message ids not processed     */
   np_rback. = 0        /* Referback on msgid to not_procd index  */
   max_cc = 0
   sum_txn. = ""        /* Summary contention txns waiting & holding */
   sum_txn.0 = 0        /* Summary count of contention txns (w & h)  */
   sum_txn_dl. = 0      /* Summary count of transaction deadlocks    */
   sum_txn_to. = 0      /* Summary count of transaction timeouts     */
   contention.  = ""    /* Contention summary for each day           */
   contention.0 = 0     /* Count of days summarized                  */
re_init:
   escalation. = ""     /* Captures lock escalation information      */
   escalation.0 = 0     /* Count of escalations captured             */
   deadlocks. = ""      /* Used to build deadlock messages           */
   deadlocks.0 = 0      /* Count of deadlock messages                */
   timeouts. = ""       /* Used to build timeout messages            */
   timeouts.0 = 0       /* Count of timeout messages                 */
   correlation. = ""    /* Correlation id used to join deadlock and  */
                        /*   timeout messages with resource unavails */
   correlation.0 = 0    /* Count of correlation ids (idx for type,   */
                        /*   connection and refback.                 */
   connection. = ""     /* Connection id for dl/to references        */
   type. = ""           /* Type of delay refd (deadlock/timeout)     */
   refback. = 0         /* deadlocks. or timeouts. refback index     */
   pass = 1             /* Used in message processing                */
   conten_obj. = ""     /* Contention objects                        */
   conten_obj.0 = 0     /* Count of contention objects               */
   conten_dl. = 0       /* Count of deadlocks on contention objects  */
   conten_to. = 0       /* Count of timeouts on contention objects   */
   con_txn. = ""        /* Contention transactions waiting & holding */
   con_txn.0 = 0        /* Count of contention transactions          */
   con_txn_dl. = 0      /* Count of transaction deadlocks            */
   con_txn_to. = 0      /* Count of transaction timeouts             */
   con_htxn. = ""       /* Contention transactions holding           */
   con_htxn.0 = 0       /* Count of contention transactions holding  */
   con_htxn_dl. = 0     /* Count of holding transaction deadlocks    */
   con_htxn_to. = 0     /* Count of holding transaction timeouts     */
```

```
   con_wtxn. = ""       /* Contention transactions waiting          */
   con_wtxn.Ø = Ø       /* Count of contention transactions waiting  */
   con_wtxn_dl. = Ø     /* Count of waiting transaction deadlocks    */
   con_wtxn_to. = Ø     /* Count of waiting transaction timeouts     */
   trc_data_lost = Ø    /* Count of lost online monitor buffers      */
   ds_ext. = ""         /* Datasets that fail to extend              */
   ds_ext.Ø = Ø         /* Count of datasets that fail to extend     */
   ds_ext_name. = ""    /* Datasets that fail to extend              */
   ds_ext_name.Ø = Ø    /* Count of datasets that fail to extend     */
   ds_ext_name_ct. = Ø  /* Count for each dataset of no of ext fails */
return
/* get_joblog ----------------------------------------------------*/
/* This procedure is used to read in the DB2 <ssid>MSTR job log.     */
/* get_joblog ----------------------------------------------------*/
get_joblog:
   if max_cc > Ø then return   /* if in trouble, then skip this one */
   address TSO
   "execio * DISKR JLØ1 (FINIS"
   if queued() = Ø then
               call bad_error("No data available in joblog dataset")
return
/* bad_error -----------------------------------------------------*/
/* This procedure is used to report error conditions in a tidy way.  */
/* bad_error -----------------------------------------------------*/
bad_error:
   say left("",79,"=")
   say " "
   say center("No data available in joblog dataset",79)
   say " "
   say left("",79,"=")
   max_cc = 8
return
/* process_joblog ------------------------------------------------*/
/* This procedure is used to collate information from the joblog.    */
/* Note that it is assumed the joblog has been read into the         */
/* stack.                                                            */
/* process_joblog ------------------------------------------------*/
process_joblog:
   first_date = true
   do while queued() > Ø
      /* Get next log message */
      pull logmsg
      /* Capture changes in day/date */
      if substr(logmsg,2Ø,4) = "----" then do
         if ¬first_date then call print_report
                        else first_date = false
         log_day = substr(logmsg,25,22)
         logdate = word(log_day,1) word(log_day,2) word(log_day,3),
                   word(log_day,4)
      end
```

```
/* Get message time and ID, when they're there */
t_ltime = substr(logmsg,2,8)
if t_ltime /= "        " then do
   logtime = strip(t_ltime)
   tmid = strip(substr(logmsg,21,8))
   if tmid /= "" then msgid = tmid
end
/* If it's a DB2 message, process it */
if substr(msgid,1,3) = "DSN" then do
   select
      when msgid = "DSNT375I" then call deadlock
      when msgid = "DSNT376I" then call timeout
      when msgid = "DSNT501I" then call res_unavailable
      when msgid = "DSNW133I" then call trace_data_lost
      when msgid = "DSNI031I" then call lock_escalation
      when msgid = "DSNP007I" then call ds_extend_failed
      when msgid = "DSNZ002I" then nop  /* DSNZPARM mod name  */
      when msgid = "DSNY001I" then nop  /* Subsystem restart  */
      when msgid = "DSNY002I" then nop  /* Subsystem stopping */
      when msgid = "DSNR001I" then nop  /* Restart initiated  */
      when msgid = "DSNR002I" then nop  /* Restart complete   */
      when msgid = "DSNR003I" then nop  /* Restart checkpoint */
      when msgid = "DSNR004I" then nop  /* Rstrt stat counts  */
      when msgid = "DSNR005I" then nop  /* Rstrt fwd rec      */
      when msgid = "DSNR006I" then nop  /* Rstrt bckwd rec    */
      when msgid = "DSNR007I" then nop  /* Rstrt stat table   */
      when msgid = "DSNI029I" then nop  /* Fast Log Apply-FLA */
      when msgid = "DSNI028I" then nop  /* FLA Rec & Buff inf */
      when msgid = "DSNL021I" then nop  /* command accepted   */
      when msgid = "DSNL003I" then nop  /* DDF Starting       */
      when msgid = "DSNL519I" then nop  /* DDF TCP/IP info    */
      when msgid = "DSNL004I" then nop  /* DDF start complete */
      when msgid = "DSNL005I" then nop  /* DDF stopping       */
      when msgid = "DSNL006I" then nop  /* DDF stop complete  */
      when msgid = "DSNL030I" then nop  /* DDF process error  */
      when msgid = "DSN9022I" then nop  /* Start DB2 complete */
      when msgid = "DSN3002I" then nop  /* Rec incomplete for */
      when msgid = "DSN3201I" then nop  /* Abnormal EOT       */
      when msgid = "DSN7407I" then nop  /* ARM failed         */
      when msgid = "DSNJ001I" then nop  /* Active log dataset */
      when msgid = "DSNJ002I" then nop  /* Log ds full        */
      when msgid = "DSNJ003I" then nop  /* Archive ds full    */
      when msgid = "DSNJ099I" then nop  /* Log rec restart    */
      when msgid = "DSNJ127I" then nop  /* BSDS timestamp     */
      when msgid = "DSNJ139I" then nop  /* Log offload ended  */
      when msgid = "DSNJ310I" then nop  /* Arch log command   */
      when msgid = "DSNJ311I" then nop  /* Arch log cmd start */
      when msgid = "DSNJ312I" then nop  /* Arch log cmd end   */
      when msgid = "DSNT377I" then nop  /* Indoubt thd conten */
      when msgid = "DSNT736I" then nop  /* STOP DB comd ok    */
```

```
            when msgid = "DSNU1122" then nop  /* util couldn't drain*/
            when msgid = "DSNV401I" then nop  /* display thread rept*/
            when msgid = "DSNV419I" then nop  /* no connections     */
            when msgid = "DSNV434I" then nop  /* no postponed aborts*/
            when msgid = "DSNW123I" then nop  /* Trace resumed on.. */
            when msgid = "DSNW131I" then nop  /* -STOP TRACE ok     */
            otherwise do
               if np_rback.msgid = 0 then do /* ie not already got  */
                  npidx = not_procd.0 + 1
                  not_procd.0 = npidx
                  not_procd.npidx = msgid
                  np_rback.msgid = npidx
               end
            end
         end /* select */
      end
   end
   delstack
return
/* deadlock ------------------------------------------------------*/
/* This procedure is used to collate information generated by the    */
/* DSNT375I deadlock message. It is correlated with the DSNT501I     */
/* message naming the resource that is locked.                       */
/* deadlock ------------------------------------------------------*/
deadlock:
   /* Which row of the message are we looking at? */
   select
      when substr(logmsg,21,8) = msgid then do  /* 1st row inc plan1 */
         dlp1 = word(substr(logmsg,41,8),1)
         pass = 1
      end
      when substr(logmsg,29,14) = "CORRELATION-ID" & pass = 1 then do
         idx = correlation.0 + 1
         correlation.0 = idx
         correlation.idx = word(substr(logmsg,44,18),1)
      end
      when substr(logmsg,29,13) = "CONNECTION-ID" & pass = 1 then do
         idx = correlation.0
         connection.idx = word(substr(logmsg,43,8),1)
         if connection.idx = "BATCH"
            then dltxn1 = "Job "correlation.idx
            else dltxn1 = "Trans "substr(correlation.idx,5,4)
         pass=2
      end
      when substr(logmsg,29,13) = "IS DEADLOCKED" then do
         dlp2 = word(substr(logmsg,53,8),1)
      end
      when substr(logmsg,29,14) = "CORRELATION-ID" & pass = 2 then do
         idx = correlation.0
         dlcorr.idx = word(substr(logmsg,44,18),1)
```

```
      end
      when substr(logmsg,29,13) = "CONNECTION-ID" & pass = 2 then do
         dlcon2 = word(substr(logmsg,43,8),1)
         if dlcon2 = "BATCH"
            then dltxn2 = "Job "dlcorr.idx
            else dltxn2 = "Trans "substr(dlcorr.idx,5,4)
         dlidx = deadlocks.Ø + 1
         idx = correlation.Ø
         deadlocks.dlidx = logtime "Plan "dlp1,
                           dltxn1 "in "connection.idx,
                           "with Plan "dlp2 dltxn2 "in "dlcon2" on"
         deadlocks.Ø = dlidx
         refback.idx = dlidx
         type.idx = "DEADLOCK"
         call contentrans word(dltxn1,2),word(dltxn2,2),"DEADLOCK"
      end
      otherwise nop
   end
return
/* timeout ------------------------------------------------------*/
/* This procedure is used to collate information generated by the    */
/* DSNT376I timeouts message. It is correlated with the DSNT5Ø1I     */
/* message naming the resource that is locked.                       */
/* timeout ------------------------------------------------------*/
timeout:
   select
      when substr(logmsg,21,8) = msgid then do
         top1 = word(substr(logmsg,41,8),1)
         pass = 1
      end
      when substr(logmsg,29,14) = "CORRELATION-ID" & pass = 1 then do
         idx = correlation.Ø + 1
         correlation.Ø = idx
         correlation.idx = word(substr(logmsg,44,18),1)
      end
      when substr(logmsg,29,13) = "CONNECTION-ID" & pass = 1 then do
         idx = correlation.Ø
         connection.idx = word(substr(logmsg,43,8),1)
         if connection.idx = "BATCH"
            then totxn1 = "Job "correlation.idx
            else totxn1 = "Trans "substr(correlation.idx,5,4)
         pass=2
      end
      when substr(logmsg,29,13) = "IS TIMED OUT." then do
         top2 = word(substr(logmsg,78,8),1)
      end
      when substr(logmsg,29,14) = "CORRELATION-ID" & pass = 2 then do
         idx = correlation.Ø
         otcorr.idx = word(substr(logmsg,44,18),1)
      end
```

```
      when substr(logmsg,29,13) = "CONNECTION-ID" & pass = 2 then do
         tocon2 = word(substr(logmsg,43,8),1)
         if tocon2 = "BATCH"
            then totxn2 = "Job "otcorr.idx
            else totxn2 = "Trans "substr(otcorr.idx,5,4)
         toidx = timeouts.Ø + 1
         idx = correlation.Ø
         timeouts.toidx = logdate logtime ,
                          "Plan ("connection.idx") "top1" is "||,
                          "timed out against ("tocon2") "top2" on "
         timeouts.toidx = logtime "Plan "top1,
                          totxn1 "in "connection.idx,
                          "to Plan "top2 totxn2 "in "tocon2" on"
         timeouts.Ø = toidx
         refback.idx = toidx
         type.idx = "TIMEOUT"
         call contentrans word(totxn1,2),word(totxn2,2),"TIMEOUT"
      end
      otherwise nop
   end
return
/* res_unavailable ------------------------------------------------*/
/* This procedure is used to capture DSNT5Ø1I resource unavailable   */
/* messages and correlate them with their originators.               */
/* res_unavailable ------------------------------------------------*/
res_unavailable:
   select
      when substr(logmsg,32,14) = "CORRELATION-ID" then
         rucorr = strip(substr(logmsg,47,18))
      when substr(logmsg,32,13) = "CONNECTION-ID" then
         ruconn = strip(substr(logmsg,46,8))
      when substr(logmsg,32,5) = "TYPE " then
         rutype = strip(substr(logmsg,38,8))
      when substr(logmsg,32,5) = "NAME " then do
         runame = strip(substr(logmsg,37,28))
         if correlation.Ø > Ø then do
            do i = 1 to correlation.Ø
               if correlation.i = rucorr & connection.i = ruconn then do
                  cont_name = get_object_name(rutype,runame)
                  if type.i = "DEADLOCK" then do
                     dlidx = refback.i
                     deadlocks.dlidx = deadlocks.dlidx||" "cont_name
                     call contention word(cont_name,2),"DEADLOCK"
                  end
                  else if type.i = "TIMEOUT" then do
                     toidx = refback.i
                     timeouts.toidx = timeouts.toidx||" "cont_name
                     call contention word(cont_name,2),"TIMEOUT"
                  end
                  correlation.i = "#############"
```

```
                  connection.i = "#########"
                  type.i = "#########"
                  refback.i = Ø
                  leave i
               end
            end
         end
      end
      otherwise nop
   end
return
/* trace_data_lost ---------------------------------------------------*/
/* This procedure is used to track the number of times that the      */
/* online monitoring tool lost trace data as a result of output      */
/* buffer overwrites. It is reported on a daily basis.               */
/* trace_data_lost ---------------------------------------------------*/
trace_data_lost:
   trc_data_lost = trc_data_lost + 1
return
/* lock_escalation ---------------------------------------------------*/
/* This procedure tallies the number of times that a particular      */
/* object experiences a lock escalation condition.                   */
/* lock_escalation ---------------------------------------------------*/
lock_escalation:
   select
      when substr(logmsg,23,15) = "RESOURCE NAME =" then do
         objname = strip(substr(logmsg,39,28))
      end
      when substr(logmsg,23,12) = "LOCK STATE =" then do
         objlock = strip(substr(logmsg,37,8))
      end
      when substr(logmsg,23,9) = "PLAN NAME" then do
         objplan = strip(substr(logmsg,5Ø,8))
         objpack = strip(substr(logmsg,61,8))
      end
      when substr(logmsg,23,14) = "CORRELATION-ID" then do
         objcorr = strip(substr(logmsg,4Ø,16))
      end
      when substr(logmsg,23,13) = "CONNECTION-ID" then do
         objconn = strip(substr(logmsg,39,16))
         objidx = escalation.Ø + 1
         escalation.Ø = objidx
         escalation.objidx = left(logtime,1Ø)" "||,
                             left(objname,28)" "||,
                             left("'"||objlock||"'",5)" "||,
                             left(objconn,12)||" "||,
                             left(objcorr,16)||" "||,
                             left(objplan,8)||"  "||objpack
      end
      otherwise nop
```

```
   end
return
/* ds_extend_failed ------------------------------------------------*/
/* This procedure tallies the number of times that a particular      */
/* object experiences a dataset extend failure.                      */
/* ds_extend_failed ------------------------------------------------*/
ds_extend_failed:
   select
      when substr(logmsg,21,8) = msgid then nop
      when substr(logmsg,32,3) = "RC=" then nop
      when substr(logmsg,32,13) = "CONNECTION-ID" then do
         dsesplit = translate(substr(logmsg,32,54)," ","=")
         dsesplit = translate(dsesplit," ",",")
         dseconn = word(dsesplit,2)
         dsecorr = word(dsesplit,4)
         dseidx = ds_ext.0 + 1
         ds_ext.0 = dseidx
         ds_ext.dseidx = left(dseconn,8)" "||,
                         left(dsecorr,16)" "||,
                         left(dsedsn,54)
         found = false
         if ds_ext_name.0 > 0 then do
            do i = 1 to ds_ext_name.0
               if dsedsn = ds_ext_name.i then do
                  ct = ds_ext_name_ct.i + 1
                  ds_ext_name_ct.i = ct
                  found = true
               end
            end
         end
         if ¬found then do
            ct = ds_ext_name.0 + 1
            ds_ext_name.ct = dsedsn
            ds_ext_name_ct.ct = 1
            ds_ext_name.0 = ct
         end
      end
      when substr(logmsg,32,7) = "LUW-ID=" then nop
/*    DSNP007I .DSNP DSNPXTN0 - EXTEND FAILED FOR
                  DSNP.DSNDBD.DSNDB07.DSN4K18.I0001.A001.
                 RC=00D70027
                  CONNECTION-ID=BATCH, CORRELATION-ID=OPSCEUD5,
                 LUW-ID=*  */
      otherwise do
         dsedsn = strip(substr(logmsg,32,54))
      end
   end
return
/* get_object_name -------------------------------------------------*/
/* This function takes the diagnostic object type and name from the  */
```

```
/* resource unavailable message and turns it into a real object      */
/* name - usually a table or index and page number.                   */
/* get_object_name ---------------------------------------------------*/
get_object_name:
   arg rutp,runm
   select
      when rutp = "00000302" then do   /* tablespace page */
         obnm = translate(runm," ",".")
         dbnm = word(obnm,1)
         tsnm = word(obnm,2)
         pgnm = word(obnm,3)
         objname = "Tablespace "dbnm"."tsnm" page "pgnm
      end
      otherwise objname = "("rutp") "runm
   end
return objname
/* contentrans -------------------------------------------------------*/
/* This procedure is used to collate all of the contention jobs or   */
/* transactions and how many times they are involved in contention.  */
/* contentrans -------------------------------------------------------*/
contentrans:
   arg txn1,txn2,cont_type
   found = false
   if con_wtxn.0 > 0 then do
      do j = 1 to con_wtxn.0
         if con_wtxn.j = txn1 then do
            found = true
            if cont_type = "DEADLOCK" then
                 con_wtxn_dl.j = con_wtxn_dl.j + 1
            else con_wtxn_to.j = con_wtxn_to.j + 1
            leave j
         end
      end
   end
   if (con_wtxn.0 = 0) | ¬found then do
      conidx = con_wtxn.0 + 1
      con_wtxn.0 = conidx
      con_wtxn.conidx = txn1
      if cont_type = "DEADLOCK" then
           con_wtxn_dl.conidx = con_wtxn_dl.conidx + 1
      else con_wtxn_to.conidx = con_wtxn_to.conidx + 1
   end
   found = false
   if con_htxn.0 > 0 then do
      do j = 1 to con_htxn.0
         if con_htxn.j = txn2 then do
            found = true
            if cont_type = "DEADLOCK" then
                 con_htxn_dl.j = con_htxn_dl.j + 1
            else con_htxn_to.j = con_htxn_to.j + 1
```

```
            leave j
         end
      end
   end
   if (con_htxn.Ø = Ø) | ¬found then do
      conidx = con_htxn.Ø + 1
      con_htxn.Ø = conidx
      con_htxn.conidx = txn2
      if cont_type = "DEADLOCK" then
           con_htxn_dl.conidx = con_htxn_dl.conidx + 1
      else con_htxn_to.conidx = con_htxn_to.conidx + 1
   end
   found = false
   if con_txn.Ø > Ø then do
      do j = 1 to con_txn.Ø
         if con_txn.j = left(txn1,9) txn2 then do
            found = true
            if cont_type = "DEADLOCK" then
                 con_txn_dl.j = con_txn_dl.j + 1
            else con_txn_to.j = con_txn_to.j + 1
            leave j
         end
      end
   end
   if (con_txn.Ø = Ø) | ¬found then do
      conidx = con_txn.Ø + 1
      con_txn.Ø = conidx
      con_txn.conidx = left(txn1,9) txn2
      if cont_type = "DEADLOCK" then
           con_txn_dl.conidx = con_txn_dl.conidx + 1
      else con_txn_to.conidx = con_txn_to.conidx + 1
   end
   found = false
   if sum_txn.Ø > Ø then do
      do j = 1 to sum_txn.Ø
         if sum_txn.j = left(txn1,9) txn2 then do
            found = true
            if cont_type = "DEADLOCK" then
                 sum_txn_dl.j = sum_txn_dl.j + 1
            else sum_txn_to.j = sum_txn_to.j + 1
            leave j
         end
      end
   end
   if (sum_txn.Ø = Ø) | ¬found then do
      sumidx = sum_txn.Ø + 1
      sum_txn.Ø = sumidx
      sum_txn.sumidx = left(txn1,9) txn2
      if cont_type = "DEADLOCK" then
           sum_txn_dl.sumidx = sum_txn_dl.sumidx + 1
```

```
      else sum_txn_to.sumidx = sum_txn_to.sumidx + 1
   end
return
/* contention ------------------------------------------------------*/
/* This procedure is used to collate all the contention objects,     */
/* and how many times they are involved in contention issues.        */
/* contention ------------------------------------------------------*/
contention:
   arg objname,cont_type
   found = false
   if conten_obj.0 > 0 then do
      do j = 1 to conten_obj.0
         if conten_obj.j = objname then do
            found = true
            if cont_type = "DEADLOCK" then
                 conten_dl.j = conten_dl.j + 1
            else conten_to.j = conten_to.j + 1
            leave j
         end
      end
   end
   if (conten_obj.0 = 0) | ¬found then do
      conidx = conten_obj.0 + 1
      conten_obj.0 = conidx
      conten_obj.conidx = objname
      if cont_type = "DEADLOCK" then
           conten_dl.conidx = conten_dl.conidx + 1
      else conten_to.conidx = conten_to.conidx + 1
   end
return
/* print_report ----------------------------------------------------*/
/* This procedure prints a report of the information captured on a   */
/* day by day basis. It is called each time the date changes in the  */
/* joblog.                                                           */
/* print_report ----------------------------------------------------*/
print_report:
   say " "
   say " "
   say "REPORT FOR "logdate
   say " "
   day = contention.0 + 1
   contention.0 = day
   contention.day = left(log_day,24) right(deadlocks.0,5),
                    right(timeouts.0,8)
   if conten_obj.0 > 0 then do
      say "CONTENTION REPORT - OBJECTS"
      say left("Object",30) "DLocks" "TOuts"
      say left("======",30) "======" "=====""
      do i = 1 to conten_obj.0
         say left(conten_obj.i,30) right(conten_dl.i,5),
```

```
            right(conten_to.i,5)
      end
      say " "
      say "CONTENTION REPORT - TRANSACTIONS/JOBS"
      say "Victim    DLocks    TOuts"
      say "======    ======    =====""
      do i = 1 to con_wtxn.Ø
         say left(con_wtxn.i,1Ø),
             right(con_wtxn_dl.i,4) right(con_wtxn_to.i,8)
      end
      say " "
      say "Holder    DLocks    TOuts"
      say "======    ======    =====""
      do i = 1 to con_htxn.Ø
         say left(con_htxn.i,1Ø),
             right(con_htxn_dl.i,4) right(con_htxn_to.i,8)
      end
      say " "
      say "Victim    Holder    DLocks    TOuts"
      say "======    ======    ======    =====""
      do i = 1 to con_txn.Ø
         say left(con_txn.i,2Ø),
             right(con_txn_dl.i,4) right(con_txn_to.i,8)
      end
   end
   if deadlocks.Ø > Ø then do
      say " "
      say "Deadlocks"
      say "=========="
      do i = 1 to deadlocks.Ø
         say deadlocks.i
      end
   end
   if timeouts.Ø > Ø then do
      say " "
      say "Timeouts"
      say "========"
      do i = 1 to timeouts.Ø
         say timeouts.i
      end
   end
   if trc_data_lost > Ø then do
      say " "
      say "Trace data was missed by the online monitor "trc_data_lost,
          "time(s)."
   end
   if escalation.Ø > Ø then do
      say " "
      say "Lock escalation occurred "escalation.Ø" time(s):"
      say left(" Time   ",1Ø)" "||,
```

```
          left("Object locked",28)" "||,
          left("Lock",5)" "||,
          left("Connection",12)||" "||,
          left("Correlation",16)||" "||,
          left("Plan",8)||"  Package"
       say left("========",1Ø)" "||,
          left("=============",28)" "||,
          left("====",5)" "||,
          left("==========",12)||" "||,
          left("===========",16)||" "||,
          left("====",8)||"  ======="
       do i = 1 to escalation.Ø
          say escalation.i
       end
    end
    if ds_ext.Ø > Ø then do
       say " "
       say "Datasets failed to extend "ds_ext.Ø" time(s):"
/*     say left("Conn",8)" "left("Corr",16)" "left("Dataset",54)     */
/*     say left("====",8)" "left("====",16)" "left("=======",54)     */
/*     do i = 1 to ds_ext.Ø                                          */
/*        say ds_ext.i                                               */
/*     end                                                           */
       say left("Dataset",54)" "left("Count",5)
       say left("=======",54)" "left("=====",5)
       do i = 1 to ds_ext_name.Ø
          say left(ds_ext_name.i,54)" "right(ds_ext_name_ct.i,5)
       end
    end
    call re_init
return
/* report_summary ------------------------------------------------------*/
/* This procedure is used to report a summary of all contentions      */
/* (ie the number of deadlocks and timeouts each day)                 */
/*  report_summary-----------------------------------------------------*/
report_summary:
    say " "
    say " "
    say "CONTENTION REPORT - SUMMARY "
    say " "
    say "Victim    Holder    DLocks    TOuts"
    say "======    ======    ======    =====""
    do i = 1 to sum_txn.Ø
       say left(sum_txn.i,2Ø),
           right(sum_txn_dl.i,4) right(sum_txn_to.i,8)
    end
    say " "
    say "                          Dlocks    TOuts"
    say "                          ======    =====""
    Do i = 1 to contention.Ø
```

```
      say contention.i
   end
   say " "
return
/* report_others ------------------------------------------------------*/
/* This procedure is used to report all the message ids that were    */
/* produced by DB2, but not processed by this EXEC.                  */
/* report_others ------------------------------------------------------*/
report_others:
   if not_procd.Ø > Ø then do
      say " "
      say "DB2 MESSAGE IDS NOT PROCESSED:"
      do i = 1 to not_procd.Ø
         say not_procd.i
      end
   end
return
error:
   do while queued() > Ø
      pull thing
   end
   delstack
exit 8
/*--------------------------------------------------------------------*/
/* Appendix 1 - Messages Supported                                    */
/* ===============================                                    */
/* This section completes the documentation of this code. It          */
/* provides a list of all the DB2 console messages that are           */
/* supported by the EXEC.                                             */
/* Msg       Purpose                                                  */
/* ===       =======                                                  */
/* DSNT375I  Deadlock notification message.                           */
/* DSNT376I  Timeout notification message.                            */
/* DSNT5Ø1I  Resource unavailable notification.                       */
/* DSNIØ31I  Lock escalation message.                                 */
/* DSNPØØ7I  Dataset failed to extend.                                */
DSNT375I PLAN=plan-id1 WITH CORRELATION-ID=id1 CONNECTION-ID=id2
       LUW-ID=id3 THREAD-INFO=id4 IS DEADLOCKED WITH PLAN=plan-id2
       WITH CORRELATION-ID=id5 CONNECTION-ID=id6 LUW-ID=id7
       THREAD-INFO=id8 ON MEMBER id9
DSNT376I PLAN=plan-id1 WITH CORRELATION-ID=id1 CONNECTION-ID=id2
       LUW-ID=id3 THREAD-INFO=id4 IS TIMED OUT. ONE HOLDER OF THE
       RESOURCE IS PLAN=plan-id2  WITH CORRELATION-ID=id5
       CONNECTION-ID=id6 LUW-ID=id7 THREAD-INFO= id8 ON MEMBER id9
DSNT5Ø1I csect-name RESOURCE UNAVAILABLE
       CORRELATION-ID=id1 CONNECTION-ID=id2 LUW-ID=id3
       REASON reason TYPE type NAME name
DSNIØ31I  csect-name - LOCK ESCALATION HAS OCCURRED FOR
       RESOURCE NAME = name, LOCK STATE = state, PLAN NAME : PACKAGE
       NAME = id1 : id2, STATEMENT NUMBER = id3, CORRELATION-ID = id4,
```

```
       CONNECTION-ID = id5, LUW-ID = id6, THREAD-INFO = id7:id8:id9:id10
DSNP007I csect - EXTEND FAILED FOR data-set-name.  RC=rrrrrrrr
       CONNECTION-ID=xxxxxxxx, CORRELATION-ID=yyyyyyyyyyyy,
       LUW-ID=logical-unit-of-work-id =token
```

CONCLUSION

This simple report can help identify some performance and application concurrency problems. Hopefully your DB2 systems will have very few timeouts, deadlocks, lock escalation, or dataset extend problems!

Ron Brown
Principal Consultant
Triton Consulting (Germany)

DB2 Information Integrator: a federated solution

INTRODUCTION

Information integration is a collection of technologies that combines database management systems, Web services, replication, federated systems, and warehousing functions into a common platform. It also includes a variety of programming interfaces and data models. Using the information integration technology, we can access diverse types of data (structured, unstructured, and semi-structured). We can transform the data into a format that provides easy access to information across the enterprise.

DB2 Information Integrator merges diverse types of data into a format that provides easy access to information across an enterprise. With DB2 Information Integrator we can perform the following tasks:

- Access traditional forms of data and emerging data sources.
- Use data that is structured, semi-structured, and unstructured.
- Retrieve, update, transform, and replicate information from diverse distributed sources as if it were a single database regardless of where it resides.
- Centralize information to meet high-availability or performance requirements.
- Leverage on-demand access to distributed data that must remain in place.
- Integrate diverse and distributed data, without moving the data or changing the platforms.
- Work within your existing infrastructure.
- Combine existing data and content assets in new ways.
- Deliver correlated information more quickly.
- Speed time to value for new composite applications.
- IBM experiments reveal that by using DB2 Information Integrator instead of hand-coding direct access to diverse sources hand-coding can be reduced by 40%–65%.
- Reduce skill requirements typically required to decompose and optimize queries. Reduce overall development time by half.
- Reduce payroll costs, with more productive development.
- Reduce the need to modify or replace systems to make them work together.
- Reduce the need to manage more redundant data.

INFORMATION INTEGRATOR – RELATIONAL FEDERATED TECHNOLOGIES

Federated database management systems offer help in

accessing and manipulating disparate data. Federated systems provide a single-site image of distributed data that is stored or generated in a variety of formats. Federated systems offer a common interface for accessing this data. The federated database management system of DB2 Information Integrator features an extended catalog. This catalog is capable of maintaining statistics about remote data and using these statistics to globally optimize data access. By using the federated systems technology, you can gather statistics about remote data sources and use this information in the query optimizer technology that is provided by DB2 Universal Database. This allows you to access data quickly. Federated systems use the query rewrite facility of DB2 Universal Database to rewrite slower performing queries into their faster equivalent form.

In a federated database engine, you access data sources through software components that are called wrappers. Each wrapper contains information about the data source, such as the default mappings between the data source data types and the DB2 data types. To implement a wrapper, the server uses routines that are stored in a library called a wrapper module. These routines allow the server to perform operations such as connecting to a data source and iteratively retrieving data from it. To use the wrappers to query, retrieve, and manipulate data, you must install them, then register each one to add it to your federated system. DB2 Information Integrator federated technology provides a virtual database for multiple data sources. The data sources can:

- Run on different hardware and different operating system platforms.
- Be provided by different vendors.
- Use different application programming interfaces and different SQL dialects.

The federated systems technology enables programmers to customize their database management system to access a data source of their choosing, whether relational or non-relational.

Relational federated wrappers can query data in other Relational Database Management Systems (RDBMS). Relational and non-relational wrappers provide transparent access to these other database systems by mapping these sources to DB2. You can access these data sources with a single query and make use of the performance techniques and query rewrite functionality that is provided by federated systems and DB2.

Among the relational wrappers that are offered by the federated relational technology are wrappers for the non-IBM relational databases and Informix databases. You need the relational wrappers if you want to access data that is stored in some of the following data sources:

- Oracle
- Sybase
- Microsoft SQL Server
- IBM DB2 family of products
- Teradata
- Open Database Connectivity (ODBC) sources.

A COMPARATIVE ANALYSIS OF A FEDERATED SYSTEM AND A CONVENTIONAL SYSTEM

A comparative analysis of a federated system and a conventional system is shown in Figure 1.

A WALK THROUGH A BASIC FEDERATED QUERY

At a high level, the basic steps in a federated query are:

1 User or application submits a query.

2 Federated Server decomposes the query by source.

3 Federated Server and wrappers collaborate on a query plan.

DB2 Federated System	*Conventional system*	*Remarks*
For the user it requires a single connection to query multiple databases at different locations.	It requires multiple connections to each database.	It requires direct access for the conventional system.
Users can join tables from two or more databases of the same type or different types. For the user, it appears as a single collective database.	Users cannot join tables from multiple databases. It appears as multiple databases.	A Federated System can join tables from Oracle and DB2 or any other RDBMS databases.
From a DB2 Client, a user can not only query the DB2 family of products but also other databases like Oracle, Sybase, or Informix in DB2's own SQL.	From a DB2 Client, a user can access only the DB2 family of products.	
In a special mode called pass-thru mode, the user can use the data source's own SQL dialect to retrieve data from the data sources.	User need to use only DB2's SQL dialect.	
Connection overhead is less.	Connection overhead is more.	
Better performance and availability to query multiple databases.	Performance is an issue to access multiple databases.	
Depending on the query and the optimizer's cost model, the query can be evaluated partly at the data sources and partly at the Federated System.	The full query will be evaluated at the data source, so it is fully dependent on the data source's CPU and I/O capability and other considerations.	
The cost of transmitting data or messages between a Federated Server and data sources is taken into consideration while calculating the total cost to arrive at an optimal access path.	The cost of transmitting of data is not taken into account so there could be a network traffic problem while transferring data between data sources and the DB2 Client machine.	

Figure 1: Comparison of Federated and conventional systems

4 Federated Server implements the plan through query execution.

5 Wrappers take sources through each source's API.

6 Sources return data to the wrappers.

7 Wrappers return the data to Federated Server.

8 Federated Server compensates for work that the data sources are unable to do and combines data from different sources.

9 Federated Server returns data to the user or application.

AN EXAMPLE OF USING INFORMATION INTEGRATOR

The Bank, a small banking institution serving customers, has acquired Financial Services Inc, a regional investment services firm. Financial Services Inc has financial services that complement The Bank's existing products. The Bank wants to quickly analyse the merged customer population and identify new sales opportunities.

For existing customers of The Bank, sales representatives can now offer high-yield money market accounts, personal finance and investment brokerage services, and portfolio management services. The new customers of The Bank are candidates for interest-bearing checking accounts.

Challenge

The Bank and Financial Services Inc base their respective customer information systems on different database products. This makes it difficult to create a single, integrated customer list. The Bank uses DB2 UDB, while Financial Services uses a different relational database product. The Bank wants to make efficient use of the combined strengths of each company. However, it does not want to go through a lengthy process of merging information into a single database, since legacy applications exist that need time to be ported.

The Bank and Financial Services must integrate information across technologies and different business processes to:

- Allow customers to view a single consolidated portfolio of their banking and financial services information, regardless of how the companies store the information.
- Extract competitive value from existing information.
- Increase profitability through operational efficiencies.
- Shorten business cycles (the decision-making process) and increase profitability (reduced administrative and IT infrastructure costs).
- Improve customer satisfaction through consistent customer-policy data and a single customer contact.
- Increase revenue and market share by targeting qualified customers for specific promotions and sales opportunities.

Solution

Several information integration technologies can enable The Bank to quickly integrate the newly-combined information assets, without migrating information from one database product to another.

These technologies include:

- XML, for integrating and exchanging information across the merged enterprise. For example, exchanging information from the relational sources, text documents, and information gathered from the Web. Applications can use enterprise models that are built around XML.
- Federated systems, which enable access to information from multiple sources as if those sources were a single database, easing application development.
- Web services, providing access to information across enterprise boundaries to provide the customers and The Bank with up-to-date values of their stocks.

- Text search, which enables The Bank to categorize and search across documents that are stored in a database to identify promotions or additional text information that is tailored to a specific need of the customers.

The Bank's customers can view a personal summary of their accounts, seamlessly integrated into a single view that is independent of how the information is stored or managed. Sales representatives for The Bank can view product specials and selectively retrieve lists of customers who are identified as prospects, based on specified business criteria.

Sujit K Mishra
Senior Software Engineer
IBM Global Services (India)

What failure? Leveraging the DB2 UDB V8.2 automatic client reroute facility

Typically, when a database server crashes, each client connected to that server gets a communication error (usually SQL30081N) that terminates its connection and ultimately leads to an application error appearing on the end user's screen.

Such an outage causes a ripple effect throughout the infrastructure. End users (who could be your customers) can't do their work or complete their transactions, which results in dissatisfaction and potential consumer vulnerability issues. From a database administrator's perspective, any obvious error will likely violate some Service Level Agreement (SLA) that the Information Technology (IT) department has contractually signed with a line of business.

DB2 UDB V8.2 delivers a rich set of availability-enhancing

features such as High Availability Disaster Recovery (HADR), the inclusion of log files in on-line back-up images, the Automatic Client Reroute (ACR) feature, and more.

In this article, we'll discuss the details of the new ACR feature, how it can help IT adhere to stringent SLA contracts and, more importantly, how it can be used to address end-user and consumer satisfaction issues during an outage (by hiding the outage and reconnecting to the standby database). The new DB2 UDB V8.2 ACR facility allows client applications and their respective connections to be transparently transferred to an alternative database if a connection to the primary database cannot be established. Quite simply, what you don't know won't hurt you; you may experience a delay, or have to resubmit a transaction, but your application won't expose an SQL30081N error.

You can use ACR for connection error handling as long as the DB2 UDB client and server are both at the V8.2 level or later; their maintenance levels don't have to match, however.

The ACR feature can be used in almost any high availability configuration, including:

- A partitioned or non-partitioned database cluster.
- DataPropagator (DPropR)-style replication.
- High availability clustering software, such as IBM HACMP (High Availability Cluster MultiProcessor), and so on.
- An HADR environment – ACR, in conjunction with HADR, allows a client application to continue working with minimal interruption after a failover of the database that is being accessed. For example, if the primary HADR-enabled database crashes, the DBA must manually re-establish all broken connections, so that users will be connected to the new primary database. With ACR, you can avoid this step and provide more transparency to your application.
- A DB2 Connect environment that leverages all the advantages of a Sysplex environment for data sharing

groups with failover support, as well as failover to an alternative DB2 Connect server.

SETTING UP AUTOMATIC CLIENT REROUTE

The main goal of ACR is to enable DB2 UDB for Linux, Unix, and Windows applications to recover from a loss of communication, so that the application can continue its work with minimal interruption. As the name implies, rerouting is central to this feature. However, rerouting the application to a live database server is only possible if there is an alternative location that the client connection is aware of.

Setting an alternative database server location on the primary server enables ACR to reroute to the standby system in the event of a failure. (ACR will first attempt to re-establish the connection to the failed server.) When the connection is re-established, the application receives an error message about the transaction failure, but the application can continue and handle the transaction error programmatically.

If a client application is to connect transparently to an alternative server while recovering from a loss of communication with a failed DB2 server, the alternative server's location must be specified by using the new **UPDATE ALTERNATE SERVER FOR DATABASE** command. Figure 1 shows an example of the steps that are required to catalog an alternative server.

The alternative server information is stored on the primary server and loaded into the client's cache at connection time. This provides a central management control point for cataloging the secondary server. For applications that don't use a DB2 client to connect to the primary database (for example applications using the DB2 JDBC universal driver with a Type 4 JDBC connection), the alternative server information is stored in a special register.

The **UPDATE ALTERNATE SERVER FOR DATABASE** command can also be used with an HADR-enabled database (using the HADR_REMOTE_HOST and HADR_REMOTE_SVC parameters).

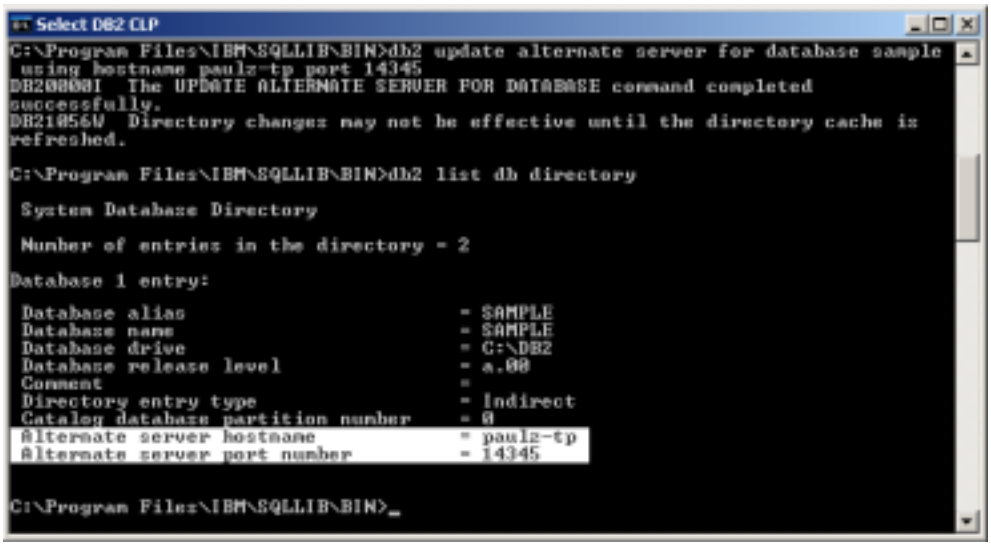

Figure 1: Steps required to catalog an alternative server

After the DBA specifies an alternative server location for a particular database and server instance, the alternative server location is returned to the client at connection time. If communication is lost for any reason, the DB2 UDB client code will be able to re-establish the connection using the alternative server information that was returned from the server (after retrying the original connection). It is important to note that because no server state is maintained across connection failures a unit of work that is in progress will have to be submitted again since it will be rolled back.

The alternative server location is maintained on the server (where it persists in the system database directory file) and on the client. The alternative server location information (host name or IP address and service name or port number) that is returned to the client at connection time is kept in the system database directory file and cached in local memory. If the alternative server location is changed on the server, the client will pick up the change at the next connection.

HOW AUTOMATIC CLIENT REROUTE WORKS

When your environment is enabled for ACR, the DB2 UDB client will retry the original server location as well as the alternative server location. This is done in case there was no real failure. For example, irregular network latency could have triggered the identification of a failed server. Retries will alternate between the original and the alternative server location for 10 minutes, or until a database connection is re-established.

If the connection is successfully re-established, SQLCODE -30108 is returned to indicate that a database connection has been re-established after a communication failure. The host name or IP address and service name or port number are also returned. The client will return the error from the original communication failure only if ACR fails to recover from it.

AUTOMATED CLIENT REROUTE – AN EXAMPLE

Figure 2 shows a typical ACR scenario:

1 The DBA creates a database and updates the alternative server location using the **UPDATE ALTERNATE SERVER FOR DATABASE** command.

2 The remote database is cataloged on the client, or in a Lightweight Directory Access Protocol (LDAP) directory.

3 The client tries to connect to the remote database.

4 After a successful connection, the alternative server location is returned to the client and saved in the system database directory and the local directory cache.

5 The application performs its work. It interacts with the database using SQL and returns result sets to the client. Then, an outage occurs during an INSERT operation. ACR checks whether or not the alternative location can be found in memory. If it is found, ACR will first retry the connection to the failed server, and if that fails it will try to connect to the alternative server.

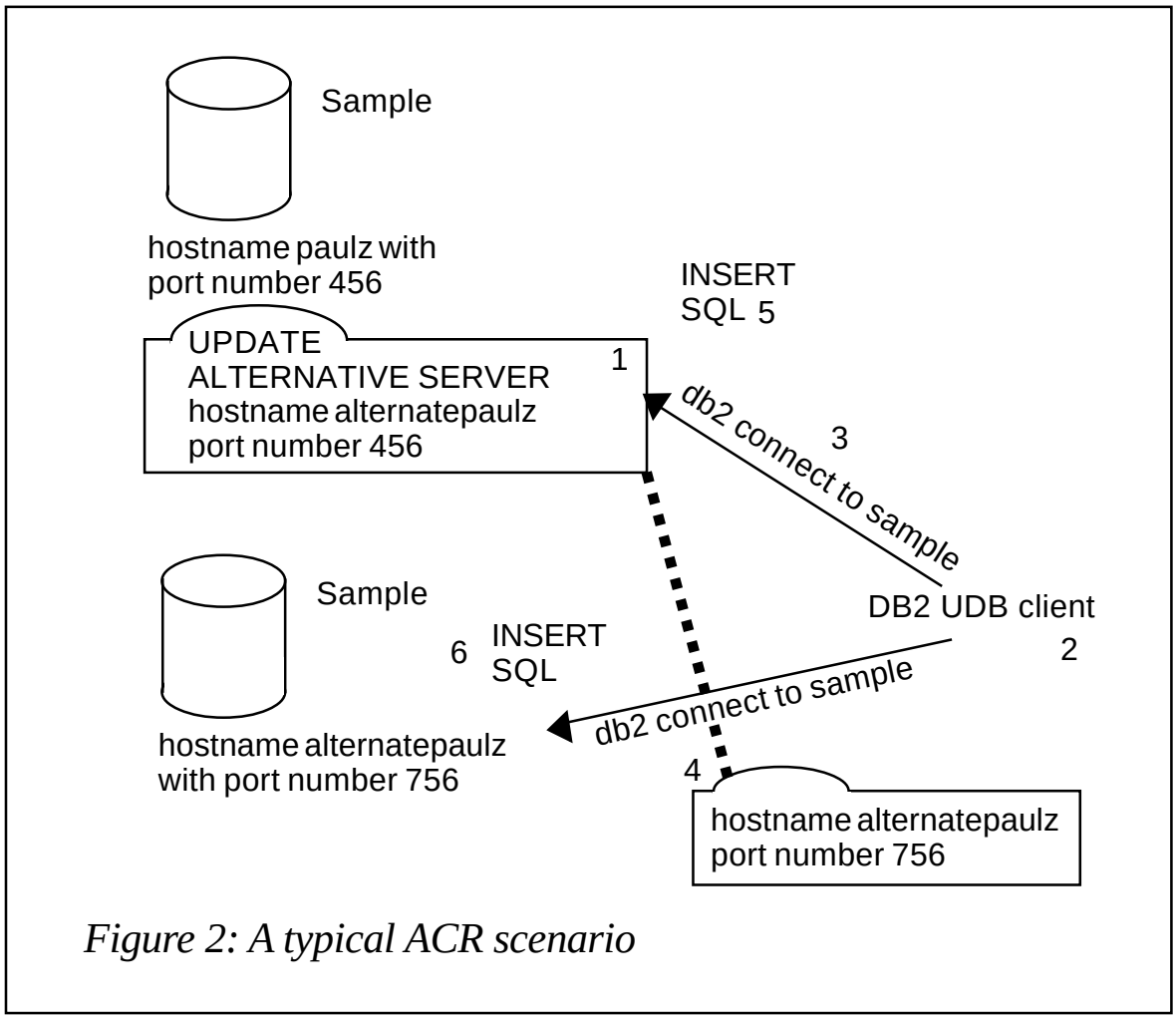

Figure 2: A typical ACR scenario

6 The client connection is re-established with the alternative server. The transaction is rolled back and then re-issued on the alternative server (assuming that this is how the application was programmed to handle this type of error).

OTHER AUTOMATIC CLIENT REROUTE CONSIDERATIONS

If you are using a JDBC driver (for example, a Type 4 driver) to connect to DB2, there is no directory file that can be populated with the alternative server's location. In the case of a Type 4 driver, the new alternateDataSource property can be used to store an alternative server's location (it contains the alternative server's JNDI location).

The alternative server information for a Type 4 JDBC driver is

dynamically copied from the server to the client at connection time and persists in the driver's static memory.

After a failover and the connection is re-established, the JDBC driver will throw a java.sql.SQLException to the application with SQLCODE -4490, which indicates that a failover has occurred and that the transaction has failed. The application can then recover and re-try the transaction.

ACR also supports LDAP. In an LDAP environment, database and node directory information is maintained at an LDAP server. Clients retrieve information from the LDAP directory. For performance reasons, customers often enable the DB2LDAPCACHE registry variable to cache this connection information in the client's local database and node directories. When using ACR with LDAP, there are some additional points that the DBA should consider:

- The **UPDATE ALTERNATE SERVER FOR DATABASE** command has been extended for an LDAP environment and you should use these extensions when cataloging the database.
- Another way to specify an alternative server's location is to use a DNS entry to specify the alternative server's IP address (you do this by specifying the multi-address hostname as the alternative server). In this scenario, the client would not know about an alternative server, but at connect time DB2 UDB would alternate between the IP addresses returned by the gethostbyname() function.

KEEPING IT ALL HIDDEN

Although IT infrastructures have matured so much that the mean time between failures is continually on the rise, failures do happen. With so much focus on the application 'experience', DBAs are faced with challenges to maintain connectivity. A database that has the ability to transparently handle outages has multiple benefits across your enterprise's IT ecosystem.

ACR is just another feature that makes DB2 UDB V8.2 a 'must-try' technology that is bound to make you smile.

Paul C Zikopoulos (paulz_ibm@msn.com)
IBM Database Competitive Technology team
Roman B Melnyk (roman_b_melnyk@hotmail.com)
DB2 Information Development team
IBM (Canada)

DB2 attachment primer

Until recently, most business application programs accessing DB2 mainframe data were coded using traditional high-level programming languages such as COBOL, PL/I, C, C++, and Assembler.

The term 'DB2 attachment' refers to the mechanism used by a mainframe application program coded in a high-level programming language to establish a connection to a DB2 DataBase Management System (DBMS).

TYPES OF DB2 ATTACHMENT AND WHY THEY ARE NEEDED

In most cases, with the exception of DB2 stored procedures executing under DB2 control, mainframe application programs execute in a separate address space DB2. These other address spaces are referred to as 'DB2 allied address spaces' and have run-time requirements that are different from those that DB2 itself has.

DB2 attachments exist for CICS, IMS, TSO, RRS, and batch (also known as CAF). Please note that once a DB2 program establishes a thread to DB2 using a specific type of attachment, it can't use any of the other ones. This restriction often comes into play when multiple programs using different attachments desire to use a common DB2 subroutine.

Both CICS and IMS are, like DB2, subsystems of the operating

system (OS/390 or z/OS) and they manage lots of non-DB2 resources. These subsystems also have the capability of running multiple concurrent DB2 programs. Because of the possibility that a Unit Of Work (UOW) started in either CICS or IMS affects more than DB2 data (such as a VSAM file or an IMS database), both CICS and IMS subsystems play the role of coordinator in managing the two-phase protocol that ensures the UOW's integrity. Both CICS and IMS attachments are distributed with the CICS or the IMS product libraries, not with the DB2 product libraries.

The TSO attachment allows an application program running in the TSO environment (either for a user's TSO address space or for a batch TSO session) to connect to DB2 and execute SQL statements. Programs written to use the TSO attachment run under the DSN TSO command processor of DB2.

The batch attachment is also referred to as the Call Attach Facility (CAF) attachment, not to be confused with executing the DSN command processor in batch mode. The CAF attachment allows a program to run directly as a TSO program (ie without the DSN command processor) or as an MVS batch program, providing greater control over the execution environment by allowing the direct trapping of returning error codes. It is also used by the DB2 stored procedures executing under DB2 control inside the XXXXSPAS address space, where XXXX is the name of the DB2 system. Note that the XXXXSPAS address space is no longer provided in DB2 Version 8.

RRS stands for Recoverable Resource Manager Services, and is the latest attachment used for DB2 stored procedures executing under the control of the operating system Work-Load Manager (WLM) system. This type of attachment allows a UOW to span resources across multiple subsystems – the operating system itself provides the services required to maintain the UOW's integrity across the multiple subsystems involved.

There are other ways to access mainframe DB2 data without using a DB2 attachment, such as non-mainframe remote

programs connecting to DB2 via various network protocols, or even mainframe programs coded in REXX and Java programming languages. These other ways of accessing DB2 data that do not use a DB2 attachment are beyond the scope of this article and will not be mentioned any further.

SQL AND DB2 ATTACHMENTS

An application program accesses DB2 data via Structured Query Language (SQL) statements. SQL is the standard language for defining and manipulating data in a relational database. Note that there are two ways of executing SQL statements – dynamically and statically. Some SQL statements can be executed in both ways, but there are some SQL statements that can be executed only in static mode.

Definitions for dynamic and static SQL are:

- Dynamic SQL statements are those that are prepared and executed within an application program while the program is executing, and the SQL statements can be programmatically changed several times during the application program's execution.
- Static SQL statements are those that are embedded within a program, and these statements are prepared before the program is executed. After being prepared, the SQL statements in the application do not change. The process to prepare an application program containing SQL statements will be described in detail in a future article. Notice that programs containing static SQL statements can be programmed to compose and execute dynamic SQL statements.

The DB2 attachments for TSO, batch, CICS, IMS, and RRS only support the execution of static SQL statements, although, as described above, dynamic SQL statements can be programmatically prepared and executed within a program using one of the DB2 attachments.

PREPARING A PROGRAM TO USE A DB2 ATTACHMENT

A high-level programming language application program containing embedded SQL statements must be 'prepared' using a sequence of steps as follows:

- Pre-compilation of the source code containing SQL statements.
- Optional translation of CICS commands into normal high-level programming language statements.
- High-level language compiler process.
- Operating system linkage editor process.
- Binding of the Data Base Request Module (DBRM) into a DB2 package and/or plan.

These steps are described in detail in IBM's DB2 *Application Programming and SQL Guide* manuals, available on the Internet at http://www-306.ibm.com/software/data/db2/zos/index.html.

Four important objects are created by this program preparation process. They are:

1 A DBRM.

2 An object code module.

3 An executable load module.

4 A DB2 package and/or DB2 plan.

Please take note that there is an alternative program preparation approach that combines the pre-compile and high-level language compilation steps into a single step by using the SQL coprocessor feature of the high-level language compilers. Additional information describing this approach can also be found on the *Application Programming and SQL guide* manuals for each DB2 version. It is recommended that you review this other approach and review the benefits and drawbacks introduced by it.

Next month we will look at how to use DB2 attachments.

Antonio Salcedo
DB2 Systems Programmer (USA)

Why not share your expertise and earn money at the same time? *DB2 Update* is looking for REXX EXECs, program code, JavaScript, etc, that experienced users of DB2 have written to make their life, or the lives of their users, easier. We are also looking for explanatory articles, and hints and tips, from experienced users.

Articles can be of any length and should be e-mailed to the editor, Trevor Eddolls, at trevore@xephon.com.

MAX Software has announced Release 3.3.0 of its data management product line. New features in 3.3.0 enable the exchange of mainframe data with non-mainframe systems or applications. These solutions provide a schedulable way to manipulate, privatize, and transform legacy application data into other data formats such as XML.

The main new feature in Release 3.3.0 is the ability to transform data from its COBOL format into portable formats. There's also enhanced support for worldwide code pages and Unicode technology that further helps maintain the integrity of the data, regardless of its geographic origin or destination.

MAX DB2/UTIL is the utility for mainframe DB2 users.

For further information contact:
MAX Software, 3609 S Wadsworth Blvd, Suite 500, Denver, CO 80235, USA.
Tel: (303) 985 1558.
URL: www.maxsoftware.com/pages/NewsSchedules.html.

* * *

Software Engineering GmbH has announced ImpactManager for DB2, which deals with the issues of the bind and rebind process of DB2 on z/OS.

Maintenance procedures usually contain built-in options to generate rebinds after RUNSTATS. These generated rebinds may have either no impact or a negative impact on performance. By integrating ImpactManager into the automated maintenance procedures, only the rebinds that improve the access paths are performed.

For further information contact:
Software Engineering GmbH, Robert-Stolz-Straße 5, 40470 Düsseldorf, Postfach 30 09 31, 40409 Düsseldorf, Germany.
Tel: (415) 834 3131.
URL: www.seg.de/englisch/products/db2/impact/impact.php.

* * *

itgain GmbH has announced Version 2.0 of Speedgain for DB2. It is basically a DB2 UDB monitoring and tuning product. It saves information on the availability, performance rate, and utilization of DB2, allowing the continual supervision and receipt of updates on the state of the database. The information collected is displayed in diagrams. Alert situations are indicated using traffic light icons. If bottlenecks are developing, the green light changes to yellow then red. Reports can be used in capacity planning.

The product runs on Linux, AIX, Windows 2000/XP, and Sun Solaris, and integrates with Tivoli for system monitoring.

For further information contact:
itgain GmbH, Vahrenwalder Str 269A, 30179 Hanover, Germany.
Tel: +49 511 9666 817.
URL: www.itgain.de/en/speedgain.html.

* * *

Responsive Systems has announced Version 8 of Buffer Pool Tool for DB2 Version 8.

This version of the software for tuning DB2 buffer pools reduces the overhead of collecting data, and the elapsed and CPU times for both producing statistics reports, and executing simulations (predicting the effect of changes).

For further information contact:
Responsive Systems, 281 Highway 79, Morganville, NJ 07751, USA.
Tel: (732) 972 1261.
URL: www.maxsoftware.com/pages/NewsSchedules.html.