120

DB2

October 2002

In this issue

update

DB2 Update

Published by

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

North American office

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

Subscriptions and back-issues

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

DB2 Update on-line

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

Editor

Trevor Eddolls

Disclaimer

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

Contributions

When Xephon is given copyright, articles published in *DB2 Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

What's new in DB2 UDB V8

IBM recently announced the availability of the open beta for DB2 UDB V8 (available from www.ibm.com/db2/v8beta), and this article gives an overview of my top choices of what I feel is of particular interest in this release. I tested this version of DB2 on a Windows 2000 laptop.

DB2 UDB Enterprise Server Edition (ESE) – to give the product its full name – is a replacement for Enterprise Edition (EE) and Enterprise Extended Edition (EEE). You therefore get one exe file if you have downloaded the beta (or 1 CD I guess!), and can specify whether you want to run ESE in non-partitioned or partitioned mode.

What is the main driving force behind this new release? Well, I would say availability – with more and more shops being 24x7, there is an ever-increasing pressure not to have any down time because of data loads, REORGs, etc. DB2 already had the capability to perform on-line/ incremental back-ups, and with V8 it now has the capability to perform on-line loads and REORGs.

When you install the product, you will find that the code now gets installed into C:\Program Files\IBM\ and not C:\Program Files\, as was the case in V7.

I am first going to talk about the new GUIs and then look at some of the key new features.

NEW GUIS

The GUIs have been rearranged and logically grouped as follows (I have included the shortcut commands (in italics) to enter from the CLP to fire up each of the GUIs):

- Command line tools – Command Center (*db2cmdctr*), Command Line Processor (*db2*), Command Window (*db2cw*).
- Development tools – Development Center (*db2dc*), Project Deployment Tool (*db2dcdpl*).

Show all user authorizations – part 2

This month we conclude the utility that brings all user authorizations and privileges from the DB2 catalog for a given userID or groupID. It automatically creates a sequential dataset whose format is <UserID>.SHOAUTH.INF, and writes the information to it.

```
 PERFORM 990000-WRITE-AUTHDD
 PERFORM A001-FETCH-C61
   THRU  A001-FETCH-C61-EXIT
   UNTIL SQLCODE NOT EQUAL 0
END-IF.
EXEC SQL CLOSE C6 END-EXEC.
STRING  '   ' DELIMITED BY '  ' INTO AUTH-RECORD.
PERFORM 990000-WRITE-AUTHDD.
STRING  'Database Privileges (Part 2): '
DELIMITED BY '     ' INTO AUTH-RECORD.
PERFORM 990000-WRITE-AUTHDD.
MOVE    '---------------------------- ' TO AUTH-RECORD.
PERFORM 990000-WRITE-AUTHDD.
EXEC SQL OPEN C6 END-EXEC.
EXEC SQL FETCH C6 INTO
 :WS-DB-GRANTOR
,:WS-DB-AUTHHOWGOT
,:WS-DB-NAME
,:WS-DB-CREATETABAUTH
,:WS-DB-CREATETSAUTH
,:WS-DB-DBADMAUTH
,:WS-DB-DBCTRLAUTH
,:WS-DB-DBMAINTAUTH
,:WS-DB-DISPLAYDBAUTH
,:WS-DB-DROPAUTH
,:WS-DB-IMAGCOPYAUTH
,:WS-DB-LOADAUTH
,:WS-DB-REORGAUTH
,:WS-DB-RECOVERDBAUTH
,:WS-DB-REPAIRAUTH
,:WS-DB-STARTDBAUTH
,:WS-DB-STATSAUTH
,:WS-DB-STOPAUTH
,:WS-DB-DATEGRANTED
END-EXEC.
IF SQLCODE < 0
 CALL 'DSNTIAR' USING SQLCA ERROR-MESSAGE ERROR-TEXT-LEN
 DISPLAY 'ERROR OCCURRED IN FETCH <DATABASE> PRIV.'
 DISPLAY ERROR-MESSAGE
 GO TO 000000-EXIT
```

```
    END-IF.
    IF SQLCODE = 100
     STRING  'There is NO DATABASE Privilege for this user'
     DELIMITED BY '  ' INTO AUTH-RECORD
     PERFORM 990000-WRITE-AUTHDD
     GO TO A001-DB-AUTH-EXIT
    END-IF.
    IF SQLCODE = 0
     STRING 'Grantor  '
            'How '
            'Name     '
            'Load ' 'Reorg ' 'Rec ' 'Repair '
            'Start ' 'Stat ' 'Stop ' 'Date   '
     DELIMITED BY '                       ' INTO AUTH-RECORD
     PERFORM 990000-WRITE-AUTHDD
     STRING '-------- '
            '-- '
            '-------- '
            '---- ' '---- ' '-- ' '------ '
            '---- ' '---- ' '---- ' '------ '
     DELIMITED BY '                       ' INTO AUTH-RECORD
     PERFORM 990000-WRITE-AUTHDD
     PERFORM A001-FETCH-C62
       THRU  A001-FETCH-C62-EXIT
       UNTIL SQLCODE NOT EQUAL 0
    END-IF.
    EXEC SQL CLOSE C6 END-EXEC.
A001-DB-AUTH-EXIT.
      EXIT.
A001-FETCH-C61.
     IF WS-DB-CREATETABAUTH = ' '
      MOVE '-' TO WS-DB-CREATETABAUTH
     ELSE
      MOVE 'T' TO WS-DB-CREATETABAUTH
     END-IF.
     IF WS-DB-CREATETSAUTH = ' '
      MOVE '-' TO WS-DB-CREATETSAUTH
     ELSE
      MOVE 'S' TO WS-DB-CREATETSAUTH
     END-IF.
     IF WS-DB-DBADMAUTH = ' '
      MOVE '-' TO WS-DB-DBADMAUTH
     ELSE
      MOVE 'D' TO WS-DB-DBADMAUTH
     END-IF.
     IF WS-DB-DBCTRLAUTH = ' '
      MOVE '-' TO WS-DB-DBCTRLAUTH
     ELSE
      MOVE 'C' TO WS-DB-DBCTRLAUTH
     END-IF.
```

```
 IF WS-DB-DBMAINTAUTH = ' '
  MOVE '-' TO WS-DB-DBMAINTAUTH
 ELSE
  MOVE 'M' TO WS-DB-DBMAINTAUTH
 END-IF.
 IF WS-DB-DISPLAYDBAUTH = ' '
  MOVE '-' TO WS-DB-DISPLAYDBAUTH
 ELSE
  MOVE 'P' TO WS-DB-DISPLAYDBAUTH
 END-IF.
 IF WS-DB-DROPAUTH = ' '
  MOVE '-' TO WS-DB-DROPAUTH
 ELSE
  MOVE 'R' TO WS-DB-DROPAUTH
 END-IF.
 IF WS-DB-IMAGCOPYAUTH = ' '
  MOVE '-' TO WS-DB-IMAGCOPYAUTH
 ELSE
  MOVE 'I' TO WS-DB-IMAGCOPYAUTH
 END-IF.
 STRING
 WS-DB-GRANTOR ' '
 WS-DB-AUTHHOWGOT '   '
 WS-DB-NAME ' '
 WS-DB-CREATETABAUTH '      '
 WS-DB-CREATETSAUTH  '     '
 WS-DB-DBADMAUTH '     '
 WS-DB-DBCTRLAUTH '      '
 WS-DB-DBMAINTAUTH '       '
 WS-DB-DISPLAYDBAUTH '     '
 WS-DB-DROPAUTH '    '
 WS-DB-IMAGCOPYAUTH '    '
 WS-DB-DATEGRANTED
 DELIMITED BY '                         ' INTO AUTH-RECORD.
 PERFORM 990000-WRITE-AUTHDD.
EXEC SQL FETCH C6 INTO
 :WS-DB-GRANTOR
,:WS-DB-AUTHHOWGOT
,:WS-DB-NAME
,:WS-DB-CREATETABAUTH
,:WS-DB-CREATETSAUTH
,:WS-DB-DBADMAUTH
,:WS-DB-DBCTRLAUTH
,:WS-DB-DBMAINTAUTH
,:WS-DB-DISPLAYDBAUTH
,:WS-DB-DROPAUTH
,:WS-DB-IMAGCOPYAUTH
,:WS-DB-LOADAUTH
,:WS-DB-REORGAUTH
,:WS-DB-RECOVERDBAUTH
```

```
   ,:WS-DB-REPAIRAUTH
   ,:WS-DB-STARTDBAUTH
   ,:WS-DB-STATSAUTH
   ,:WS-DB-STOPAUTH
   ,:WS-DB-DATEGRANTED
   END-EXEC.
A00-FETCH-C61-EXIT.
      EXIT.
A00-FETCH-C62.
    IF WS-DB-LOADAUTH = ' '
     MOVE '-' TO WS-DB-LOADAUTH
    ELSE
     MOVE 'L' TO WS-DB-LOADAUTH
    END-IF.
    IF WS-DB-REORGAUTH = ' '
     MOVE '-' TO WS-DB-REORGAUTH
    ELSE
     MOVE 'R' TO WS-DB-REORGAUTH
    END-IF.
    IF WS-DB-RECOVERDBAUTH = ' '
     MOVE '-' TO WS-DB-RECOVERDBAUTH
    ELSE
     MOVE 'V' TO WS-DB-RECOVERDBAUTH
    END-IF.
    IF WS-DB-REPAIRAUTH = ' '
     MOVE '-' TO WS-DB-REPAIRAUTH
    ELSE
     MOVE 'P' TO WS-DB-REPAIRAUTH
    END-IF.
    IF WS-DB-STARTDBAUTH = ' '
     MOVE '-' TO WS-DB-STARTDBAUTH
    ELSE
     MOVE 'T' TO WS-DB-STARTDBAUTH
    END-IF.
    IF WS-DB-STATSAUTH = ' '
     MOVE '-' TO WS-DB-STATSAUTH
    ELSE
     MOVE 'S' TO WS-DB-STATSAUTH
    END-IF.
    IF WS-DB-STOPAUTH = ' '
     MOVE '-' TO WS-DB-STOPAUTH
    ELSE
     MOVE 'P' TO WS-DB-STOPAUTH
    END-IF.
    STRING
    WS-DB-GRANTOR ' '
    WS-DB-AUTHHOWGOT '   '
    WS-DB-NAME ' '
    WS-DB-LOADAUTH '    '
    WS-DB-REORGAUTH '     '
```

```
       WS-DB-RECOVERDBAUTH '   '
       WS-DB-REPAIRAUTH '      '
       WS-DB-STARTDBAUTH '     '
       WS-DB-STATSAUTH '    '
       WS-DB-STOPAUTH '    '
       WS-DB-DATEGRANTED
       DELIMITED BY '                         ' INTO AUTH-RECORD.
       PERFORM 990000-WRITE-AUTHDD.
      EXEC SQL FETCH C6 INTO
       :WS-DB-GRANTOR
      ,:WS-DB-AUTHHOWGOT
      ,:WS-DB-NAME
      ,:WS-DB-CREATETABAUTH
      ,:WS-DB-CREATETSAUTH
      ,:WS-DB-DBADMAUTH
      ,:WS-DB-DBCTRLAUTH
      ,:WS-DB-DBMAINTAUTH
      ,:WS-DB-DISPLAYDBAUTH
      ,:WS-DB-DROPAUTH
      ,:WS-DB-IMAGCOPYAUTH
      ,:WS-DB-LOADAUTH
      ,:WS-DB-REORGAUTH
      ,:WS-DB-RECOVERDBAUTH
      ,:WS-DB-REPAIRAUTH
      ,:WS-DB-STARTDBAUTH
      ,:WS-DB-STATSAUTH
      ,:WS-DB-STOPAUTH
      ,:WS-DB-DATEGRANTED
      END-EXEC.
  A001-FETCH-C62-EXIT.
        EXIT.
***********  PACKAGE AUTH **********
  A001-PCK-AUTH.
      STRING  '   ' DELIMITED BY '  ' INTO AUTH-RECORD.
      PERFORM 990000-WRITE-AUTHDD.
      STRING  'Package Privileges: '
      DELIMITED BY '     ' INTO AUTH-RECORD.
      PERFORM 990000-WRITE-AUTHDD.
      MOVE    '----------------- ' TO AUTH-RECORD.
      PERFORM 990000-WRITE-AUTHDD.
      EXEC SQL OPEN C5 END-EXEC.
      EXEC SQL FETCH C5 INTO
       :WS-PCK-GRANTOR
      ,:WS-PCK-AUTHHOWGOT
      ,:WS-PCK-COLLID
      ,:WS-PCK-NAME
      ,:WS-PCK-BINDAUTH
      ,:WS-PCK-EXECUTEAUTH
      ,:WS-PCK-COPYAUTH
      ,:WS-PCK-TIMESTAMP
```

```
   END-EXEC.
   IF SQLCODE < Ø
    CALL 'DSNTIAR' USING SQLCA ERROR-MESSAGE ERROR-TEXT-LEN
    DISPLAY 'ERROR OCCURRED IN FETCH <PACKAGE> PRIV.'
    DISPLAY ERROR-MESSAGE
    GO TO ØØØØØØ-EXIT
   END-IF.
   IF SQLCODE = 1ØØ
    STRING  'There is NO PACKAGE Privilege for this user'
    DELIMITED BY '  ' INTO AUTH-RECORD
    PERFORM 99ØØØØ-WRITE-AUTHDD
    GO TO AØØ1-PCK-AUTH-EXIT
   END-IF.
   IF SQLCODE = Ø
    STRING 'Grantor  '
           'How ' 'Collid             '
           'Name     '
           'Bind ' 'Exec ' 'Copy ' 'Timestamp '
    DELIMITED BY '                       ' INTO AUTH-RECORD
    PERFORM 99ØØØØ-WRITE-AUTHDD
    STRING '-------- '
           '-- ' '----------------- '
           '-------- '
           '---- ' '---- ' '---- ' '-------- '
    DELIMITED BY '                       ' INTO AUTH-RECORD
    PERFORM 99ØØØØ-WRITE-AUTHDD
    PERFORM AØØ1-FETCH-C5
      THRU  AØØ1-FETCH-C5-EXIT
      UNTIL SQLCODE NOT EQUAL Ø
   END-IF.
   EXEC SQL CLOSE C5 END-EXEC.
AØØ1-PCK-AUTH-EXIT.
     EXIT.
AØØ1-FETCH-C5.
    IF WS-PCK-BINDAUTH = ' '
     MOVE '-' TO WS-PCK-BINDAUTH
    ELSE
     MOVE 'B' TO WS-PCK-BINDAUTH
    END-IF.
    IF WS-PCK-COPYAUTH = ' '
     MOVE '-' TO WS-PCK-COPYAUTH
    ELSE
     MOVE 'C' TO WS-PCK-COPYAUTH
    END-IF.
    IF WS-PCK-EXECUTEAUTH = ' '
     MOVE '-' TO WS-PCK-EXECUTEAUTH
    ELSE
     MOVE 'E' TO WS-PCK-EXECUTEAUTH
    END-IF.
    STRING
```

```
           WS-PCK-GRANTOR ' '
           WS-PCK-AUTHHOWGOT '   '
           WS-PCK-COLLID ' '
           WS-PCK-NAME ' '
           WS-PCK-BINDAUTH '    '
           WS-PCK-EXECUTEAUTH '    '
           WS-PCK-COPYAUTH '    '
           WS-PCK-TIMESTAMP
           DELIMITED BY '                         ' INTO AUTH-RECORD.
           PERFORM 990000-WRITE-AUTHDD.
          EXEC SQL FETCH C5 INTO
           :WS-PCK-GRANTOR
          ,:WS-PCK-AUTHHOWGOT
          ,:WS-PCK-COLLID
          ,:WS-PCK-NAME
          ,:WS-PCK-BINDAUTH
          ,:WS-PCK-EXECUTEAUTH
          ,:WS-PCK-COPYAUTH
          ,:WS-PCK-TIMESTAMP
          END-EXEC.
        A001-FETCH-C5-EXIT.
             EXIT.
      ***********  PLAN AUTH **********
        A001-PLAN-AUTH.
          STRING  '   ' DELIMITED BY '  ' INTO AUTH-RECORD.
          PERFORM 990000-WRITE-AUTHDD.
          STRING  'Plan Privileges: '
          DELIMITED BY '     ' INTO AUTH-RECORD.
          PERFORM 990000-WRITE-AUTHDD.
          MOVE    '---------------- ' TO AUTH-RECORD.
          PERFORM 990000-WRITE-AUTHDD.
          EXEC SQL OPEN C4 END-EXEC.
          EXEC SQL FETCH C4 INTO
           :WS-PLAN-GRANTOR
          ,:WS-PLAN-AUTHHOWGOT
          ,:WS-PLAN-NAME
          ,:WS-PLAN-BINDAUTH
          ,:WS-PLAN-EXECUTEAUTH
          ,:WS-PLAN-DATEGRANTED
          END-EXEC.
          IF SQLCODE < 0
           CALL 'DSNTIAR' USING SQLCA ERROR-MESSAGE ERROR-TEXT-LEN
           DISPLAY 'ERROR OCCURRED IN FETCH <PLAN> PRIV.'
           DISPLAY ERROR-MESSAGE
           GO TO 000000-EXIT
          END-IF.
          IF SQLCODE = 100
           STRING  'There is NO PLAN Privilege for this user'
           DELIMITED BY '  ' INTO AUTH-RECORD
```

```
     PERFORM 990000-WRITE-AUTHDD
     GO TO A001-PLAN-AUTH-EXIT
    END-IF.
    IF SQLCODE = 0
     STRING 'Grantor  '
            'How '
            'Name     '
            'Bind ' 'Exec ' 'Date   '
     DELIMITED BY '                      ' INTO AUTH-RECORD
     PERFORM 990000-WRITE-AUTHDD
     STRING '-------- '
            '-- '
            '-------- '
            '---- ' '---- ' '------'
     DELIMITED BY '                      ' INTO AUTH-RECORD
     PERFORM 990000-WRITE-AUTHDD
     PERFORM A001-FETCH-C4
       THRU  A001-FETCH-C4-EXIT
       UNTIL SQLCODE NOT EQUAL 0
    END-IF.
    EXEC SQL CLOSE C4 END-EXEC.
A001-PLAN-AUTH-EXIT.
      EXIT.
A001-FETCH-C4.
     IF WS-PLAN-NAME = ' '
      MOVE '-' TO WS-PLAN-NAME
     END-IF.
     IF WS-PLAN-BINDAUTH = ' '
      MOVE '-' TO WS-PLAN-BINDAUTH
     ELSE
      MOVE 'B' TO WS-PLAN-BINDAUTH
     END-IF.
     IF WS-PLAN-EXECUTEAUTH = ' '
      MOVE '-' TO WS-PLAN-EXECUTEAUTH
     ELSE
      MOVE 'E' TO WS-PLAN-EXECUTEAUTH
     END-IF.
     STRING
     WS-PLAN-GRANTOR ' '
     WS-PLAN-AUTHHOWGOT '   '
     WS-PLAN-NAME ' '
     WS-PLAN-BINDAUTH '    '
     WS-PLAN-EXECUTEAUTH '    '
     WS-PLAN-DATEGRANTED
     DELIMITED BY '                       ' INTO AUTH-RECORD.
     PERFORM 990000-WRITE-AUTHDD.
    EXEC SQL FETCH C4 INTO
     :WS-PLAN-GRANTOR
    ,:WS-PLAN-AUTHHOWGOT
```

```
      ,:WS-PLAN-NAME
      ,:WS-PLAN-BINDAUTH
      ,:WS-PLAN-EXECUTEAUTH
      ,:WS-PLAN-DATEGRANTED
      END-EXEC.
  A001-FETCH-C4-EXIT.
        EXIT.
***********  USE  AUTH **********
  A001-USE-AUTH.
      STRING  '   ' DELIMITED BY '  ' INTO AUTH-RECORD.
      PERFORM 990000-WRITE-AUTHDD.
      STRING  'Use  Privileges: '
      DELIMITED BY '     ' INTO AUTH-RECORD.
      PERFORM 990000-WRITE-AUTHDD.
      MOVE    '---------------- ' TO AUTH-RECORD.
      PERFORM 990000-WRITE-AUTHDD.
      EXEC SQL OPEN C3 END-EXEC.
      EXEC SQL FETCH C3 INTO
       :WS-USE-GRANTOR
      ,:WS-USE-AUTHHOWGOT
      ,:WS-USE-QUALIFIER
      ,:WS-USE-NAME
      ,:WS-USE-OBTYPE
      ,:WS-USE-USEAUTH
      ,:WS-USE-DATEGRANTED
      END-EXEC.
      IF SQLCODE < 0
       CALL 'DSNTIAR' USING SQLCA ERROR-MESSAGE ERROR-TEXT-LEN
       DISPLAY 'ERROR OCCURRED IN FETCH <USE> PRIV.'
       DISPLAY ERROR-MESSAGE
       GO TO 000000-EXIT
      END-IF.
      IF SQLCODE = 100
       STRING  'There is NO USE Privilege for this user'
       DELIMITED BY '  ' INTO AUTH-RECORD
       PERFORM 990000-WRITE-AUTHDD
       GO TO A001-USE-AUTH-EXIT
      END-IF.
      IF SQLCODE = 0
       STRING 'Grantor  '
              'How ' 'Qual     '
              'Name                '
              'Type ' 'Use ' 'Date   '
       DELIMITED BY '                        ' INTO AUTH-RECORD
       PERFORM 990000-WRITE-AUTHDD
       STRING '-------- '
              '-- ' '-------- '
              '------------------ '
              '---- ' '-- ' '------'
```

```
           DELIMITED BY '                        ' INTO AUTH-RECORD
           PERFORM 990000-WRITE-AUTHDD
           PERFORM A001-FETCH-C3
             THRU  A001-FETCH-C3-EXIT
             UNTIL SQLCODE NOT EQUAL 0
         END-IF.
         EXEC SQL CLOSE C3 END-EXEC.
       A001-USE-AUTH-EXIT.
            EXIT.
       A001-FETCH-C3.
           IF WS-USE-QUALIFIER = ' '
            MOVE '-' TO WS-USE-QUALIFIER
           END-IF.
           IF WS-USE-NAME = ' '
            MOVE '-' TO WS-USE-NAME
           END-IF.
           IF WS-USE-OBTYPE = ' '
            MOVE '-' TO WS-USE-OBTYPE
           END-IF.
           IF WS-USE-USEAUTH = ' '
            MOVE '-' TO WS-USE-USEAUTH
           END-IF.
           STRING
           WS-USE-GRANTOR ' '
           WS-USE-AUTHHOWGOT '   '
           WS-USE-QUALIFIER ' '
           WS-USE-NAME ' '
           WS-USE-OBTYPE '    '
           WS-USE-USEAUTH '   '
           WS-USE-DATEGRANTED
           DELIMITED BY '                         ' INTO AUTH-RECORD.
           PERFORM 990000-WRITE-AUTHDD.
         EXEC SQL FETCH C3 INTO
           :WS-USE-GRANTOR
         ,:WS-USE-AUTHHOWGOT
         ,:WS-USE-QUALIFIER
         ,:WS-USE-NAME
         ,:WS-USE-OBTYPE
         ,:WS-USE-USEAUTH
         ,:WS-USE-DATEGRANTED
         END-EXEC.
       A001-FETCH-C3-EXIT.
            EXIT.
      ***********  USER  AUTH **********
       A001-USER-AUTH.
         STRING  '   ' DELIMITED BY '  ' INTO AUTH-RECORD.
         PERFORM 990000-WRITE-AUTHDD.
         STRING  'User Privileges: '
         DELIMITED BY '  ' INTO AUTH-RECORD.
```

```
   PERFORM 990000-WRITE-AUTHDD.
   MOVE    '---------------- ' TO AUTH-RECORD.
   PERFORM 990000-WRITE-AUTHDD.
   EXEC SQL OPEN C2 END-EXEC.
   EXEC SQL FETCH C2 INTO
    :WS-USER-GRANTOR
   ,:WS-USER-AUTHHOWGOT
   ,:WS-USER-BINDADDAUTH
   ,:WS-USER-CREATEDBAAUTH
   ,:WS-USER-DISPLAYAUTH
   ,:WS-USER-STOPALLAUTH
   ,:WS-USER-SYSADMAUTH
   ,:WS-USER-DATEGRANTED
   END-EXEC.
   IF SQLCODE < 0
    CALL 'DSNTIAR' USING SQLCA ERROR-MESSAGE ERROR-TEXT-LEN
    DISPLAY 'ERROR OCCURRED IN FETCH <USER>  PRIV.'
    DISPLAY ERROR-MESSAGE
    GO TO 000000-EXIT
   END-IF.
   IF SQLCODE = 100
    STRING  'There is NO User Privilege for this user'
    DELIMITED BY '  ' INTO AUTH-RECORD
    PERFORM 990000-WRITE-AUTHDD
    GO TO A001-USER-AUTH-EXIT
   END-IF.
   IF SQLCODE = 0
    STRING 'Grantor  '
           'How ' 'Bind ' 'Cre '
           'Dis ' 'Stop ' 'Adm ' 'Date '
    DELIMITED BY '                        ' INTO AUTH-RECORD
    PERFORM 990000-WRITE-AUTHDD
    STRING '-------- '
           '-- ' '---- ' '-- '
           '-- ' '---- ' '-- ' '------'
    DELIMITED BY '                        ' INTO AUTH-RECORD
    PERFORM 990000-WRITE-AUTHDD
    PERFORM A001-FETCH-C2
      THRU  A001-FETCH-C2-EXIT
      UNTIL SQLCODE NOT EQUAL 0
   END-IF.
   EXEC SQL CLOSE C2 END-EXEC.
A001-USER-AUTH-EXIT.
     EXIT.
A001-FETCH-C2.
    IF WS-USER-BINDADDAUTH = ' '
     MOVE '-' TO WS-USER-BINDADDAUTH
    ELSE
     MOVE 'B' TO WS-USER-BINDADDAUTH
```

```
       END-IF.
       IF WS-USER-CREATEDBAAUTH = ' '
        MOVE '-' TO WS-USER-CREATEDBAAUTH
       ELSE
        MOVE 'C' TO WS-USER-CREATEDBAAUTH
       END-IF.
       IF WS-USER-DISPLAYAUTH = ' '
        MOVE '-' TO WS-USER-DISPLAYAUTH
       ELSE
        MOVE 'D' TO WS-USER-DISPLAYAUTH
       END-IF.
       IF WS-USER-STOPALLAUTH = ' '
        MOVE '-' TO WS-USER-STOPALLAUTH
       ELSE
        MOVE 'S' TO WS-USER-STOPALLAUTH
       END-IF.
       IF WS-USER-SYSADMAUTH = ' '
        MOVE '-' TO WS-USER-SYSADMAUTH
       ELSE
        MOVE 'A' TO WS-USER-SYSADMAUTH
       END-IF.
       STRING
       WS-USER-GRANTOR ' '
       WS-USER-AUTHHOWGOT '   '
       WS-USER-BINDADDAUTH '    '
       WS-USER-CREATEDBAAUTH '   '
       WS-USER-DISPLAYAUTH '   '
       WS-USER-STOPALLAUTH '    '
       WS-USER-SYSADMAUTH '   '
       WS-USER-DATEGRANTED
       DELIMITED BY '                        ' INTO AUTH-RECORD.
       PERFORM 990000-WRITE-AUTHDD.
       EXEC SQL FETCH C2 INTO
        :WS-USER-GRANTOR
       ,:WS-USER-AUTHHOWGOT
       ,:WS-USER-BINDADDAUTH
       ,:WS-USER-CREATEDBAAUTH
       ,:WS-USER-DISPLAYAUTH
       ,:WS-USER-STOPALLAUTH
       ,:WS-USER-SYSADMAUTH
       ,:WS-USER-DATEGRANTED
       END-EXEC.
  A001-FETCH-C2-EXIT.
         EXIT.
***********  TABLE AUTH **********
  A001-TABLE-AUTH.
     STRING  'Table Privileges: '
     DELIMITED BY '  ' INTO AUTH-RECORD.
     PERFORM 990000-WRITE-AUTHDD.
```

```
   MOVE    '--------------- ' TO AUTH-RECORD.
   PERFORM 990000-WRITE-AUTHDD.
   EXEC SQL OPEN C1 END-EXEC.
   EXEC SQL FETCH C1 INTO
    :WS-GRANTOR
   ,:WS-TCREATOR
   ,:WS-TTNAME
   ,:WS-AUTHHOWGOT
   ,:WS-ALTERAUTH
   ,:WS-DELETEAUTH
   ,:WS-INDEXAUTH
   ,:WS-INSERTAUTH
   ,:WS-SELECTAUTH
   ,:WS-UPDATEAUTH
   ,:WS-DATEGRANTED
   END-EXEC.
   IF SQLCODE < 0
    CALL 'DSNTIAR' USING SQLCA ERROR-MESSAGE ERROR-TEXT-LEN
    DISPLAY 'ERROR OCCURRED IN FETCH <TABLE> PRIV.'
    DISPLAY ERROR-MESSAGE
    GO TO 000000-EXIT
   END-IF.
   IF SQLCODE = 100
    STRING  'There is NO table Privilege for this user'
    DELIMITED BY '  ' INTO AUTH-RECORD
    PERFORM 990000-WRITE-AUTHDD
    GO TO A001-TABLE-AUTH-EXIT
   END-IF.
   IF SQLCODE = 0
    STRING 'Grantor  ' 'Creator  ' 'Table Name         '
           'How ' 'Alt ' 'Del '
           'Inx ' 'Ins ' 'Sel ' 'Upd ' 'Date '
    DELIMITED BY '                       ' INTO AUTH-RECORD
    PERFORM 990000-WRITE-AUTHDD
    STRING '-------- ' '-------- ' '------------------ '
           '-- ' '-- ' '-- '
           '-- ' '-- ' '-- ' '-- ' '--------'
    DELIMITED BY '                       ' INTO AUTH-RECORD
    PERFORM 990000-WRITE-AUTHDD
    PERFORM A001-FETCH-C1
      THRU    A001-FETCH-C1-EXIT
      UNTIL SQLCODE NOT EQUAL 0
   END-IF.
   EXEC SQL CLOSE C1 END-EXEC.
A001-TABLE-AUTH-EXIT.
      EXIT.
A001-FETCH-C1.
    IF WS-ALTERAUTH = ' '
     MOVE '-' TO WS-ALTERAUTH
```

```
ELSE
 MOVE 'A' TO WS-ALTERAUTH
END-IF.
IF WS-DELETEAUTH = ' '
 MOVE '-' TO WS-DELETEAUTH
ELSE
 MOVE 'D' TO WS-DELETEAUTH
END-IF.
IF WS-INDEXAUTH = ' '
 MOVE '-' TO WS-INDEXAUTH
ELSE
 MOVE 'X' TO WS-INDEXAUTH
END-IF.
IF WS-INSERTAUTH = ' '
 MOVE '-' TO WS-INSERTAUTH
ELSE
 MOVE 'I' TO WS-INSERTAUTH
END-IF.
IF WS-SELECTAUTH = ' '
 MOVE '-' TO WS-SELECTAUTH
ELSE
 MOVE 'S' TO WS-SELECTAUTH
END-IF.
IF WS-UPDATEAUTH = ' '
 MOVE '-' TO WS-UPDATEAUTH
ELSE
 MOVE 'U' TO WS-UPDATEAUTH
END-IF.
STRING
WS-GRANTOR ' ' WS-TCREATOR ' ' WS-TTNAME-TEXT ' '
WS-AUTHHOWGOT '   ' WS-ALTERAUTH '   '
WS-DELETEAUTH '   '
WS-INDEXAUTH '   ' WS-INSERTAUTH '   '
WS-SELECTAUTH '   ' WS-UPDATEAUTH '   '
WS-DATEGRANTED
DELIMITED BY '                        ' INTO AUTH-RECORD.
PERFORM 990000-WRITE-AUTHDD.
MOVE SPACE TO WS-TTNAME-TEXT ;
EXEC SQL FETCH C1 INTO
 :WS-GRANTOR
,:WS-TCREATOR
,:WS-TTNAME
,:WS-AUTHHOWGOT
,:WS-ALTERAUTH
,:WS-DELETEAUTH
,:WS-INDEXAUTH
,:WS-INSERTAUTH
,:WS-SELECTAUTH
,:WS-UPDATEAUTH
```

```
    ,:WS-DATEGRANTED
    END-EXEC.
A001-FETCH-C1-EXIT.
      EXIT.
990000-WRITE-AUTHDD.
   INSPECT AUTH-RECORD REPLACING ALL LOW-VALUES  BY  SPACES.
   WRITE AUTHDD-REC.
   MOVE SPACE TO AUTH-RECORD.
990000-WRITE-AUTHDD-EXIT.
      EXIT.
```

Mehmet Cuneyt Goksu
DB2 Specialist (Turkey)

Monitoring DataPropagator on MVS

DataPropagator is the cornerstone of IBM's data replication solutions for relational databases. It is used to propagate changes in source data to target tables on a different DB2 subsystem. To monitor that propagation process one ideally would like to track the number of changes being propagated and how long the lag is before they reach their target tables. And whenever there is a failure in the propagation, it should be reported so that some corrective action can be taken.

We are running data propagation from DB2 on one MVS system to another DB2 on a different MVS system, and we wanted a monitor which ideally would also run on MVS. However, IBM doesn't offer such an MVS monitor for DataPropagator. So here's a simple one that I wrote.

OVERVIEW

The monitor runs as a started task or batch job on an MVS system with a DB2 subsystem that has either the source tables for propagation, or the DataPropagator control tables. A report is produced showing a few control values for the particular propagation 'subscription set'; every few minutes the number of changes and the amount of lag in the *Capture* and *Apply* processes are added to the end of the report. Here

is a sample of part of a report, to show how it looks:

```
           Capture     ---- Apply ----          -- Last Synchpoint --
  Time        Lag   Cycles Changes    Lag    Lag   Time      Date
10.55.45        7     81   15053        4      7   10.55.24  25.04.2002
11.00.46        3     75   27684       13     16   11.00.16  25.04.2002
11.05.47       16     76   28370       15     16   11.05.17  25.04.2002
11.10.47        7     75   28853       17     21   11.10.13  25.04.2002
11.15.48        6     76   23737        2      6   11.15.29  25.04.2002
```

The full report starts with an explanation of each report column's meaning and the expected values when propagation is OK. The system time differences between DB2 subsystems are taken into account when calculating these lag values.

If the propagation is working correctly, the monitor just builds its report throughout the day and nobody needs to check anything.

However, if the propagation lag is more than a specified time, the monitor does some basic analysis of the excessive lag details to determine the cause (if possible). It can write warning messages to the MVS operator console and selected TSO userids, to alert the people responsible to the problem. An MVS operator (or an MVS automation product) can then react to any warning messsage.

TECHNICAL SUMMARY

The monitor runs a batch TSO step executing the REXX program DPROPMON. That invokes the PL/I program DPRMON to gather information from the DataPropagator DB2 tables on both the source and control DB2 subsystems, then it invokes the Assembler program WAITA to wait for a specified time before querying the tables again. The querying and waiting repeats continuously until the started task is stopped by a cancel command, or it can end automatically if it reaches a user-specified 'end time'.

The DPRMON program writes the data from DB2 into a temporary file that the DPROPMON program can read, check, and write at the end of its report. In case of any SQL error, the full SQLCA information is formatted by DSNTIAR to provide the message text, and that is written into the report.

All bad messages in Syslog from the monitor start with either the characters “+DpropMon: Abending” or “+DpropMon: Warning”, so they can be easily recognized by an MVS automation tool.

The monitor works with DataPropagator Versions 6 and 7, and it should work with Version 5 too. It runs with DB2 Versions 6 and 7. (To run it with DB2 Version 5 the SQL statement with ‘CASE’ in DPRMON program must be split into two and the ‘UPPER’ function removed from three SQL statements.)

USAGE NOTES

The monitor is started on a DB2 subsystem with source or control tables for propagation. It then uses a DDF connection to the other DB2 subsystem to read the rest of the DataPropagator tables (as is used by DataPropagator itself).

A separate monitor is started for each subscription set that you want to check. This allows you to specify the best parameters for each particular set, and the monitors run independently of each other.

Some part of the DataPropagator process must be in DB2 on an MVS system. If part of the process is on another platform the monitor’s report should still show correct information, assuming the monitor is able to connect from MVS to the other server.

This monitoring will detect ALL types of propagation problems, and it does not depend on recognizing particular messages. That is important because it’s possible for propagation to stop with no error message:

- Example 1 – in the case of some (rare) communication problems, *Apply* goes into a wait state and must be cancelled and forced from MVS.
- Example 2 – DB2 gets to the end of the last active log and is unable to archive; no more updates are allowed, stopping propagation.
- Example 3 – *Capture* COLD started and *Apply* was started too soon, so the *Apply* does an initial refresh then no further propagation. And if either *Capture* or *Apply* is stopped or abends, the monitor will detect that too.

For maximum propagation performance IBM recommends that *Capture* should run with no 'pruning' of CD and UOW tables (or you should use your own external pruning), because *Capture* stops reading the DB2 logs while it is pruning, thereby delaying propagation. This monitor was developed on a system with no pruning, and assumes that log reading should be continuous. Note that the next version of DataPropagator (Version 8) will do this pruning in parallel in a separate subtask.

Note that this monitor can only warn that a problem exists. The first place to check for problems is the messages in the started tasks for the *Capture*, *Apply*, and DB2 masters. Warning: the DataPropagator error messages usually include an undocumented error code, so the exact reason is not always clear. In the case of communication problems it is sometimes very difficult to determine what is wrong. Otherwise, to check a propagation problem in more depth it is often useful to use a normal DB2 monitor to check what dynamic SQL statements are active for *Apply* or *Capture* tasks. With a little experience, you should then be able to recognize what it is doing.

MONITOR START PARAMETERS

The monitor start parameters are:

- SSID – ssid of the local DB2 subsystem for the monitor to use, or it can be a group id for a DB2 data sharing group.
- QUAL – name of the *Apply* qualifier to be monitored; if it is not specified, try the DB2 ssid name (the DJRA default).
- SUBSET – name of the subscription set to be monitored; if it is not specified, try the DB2 ssid name (the DJRA default).
- APPLY – name of the *Apply* task if it is running on the same MVS or in the same JES3 multisystem complex.
- CAPT – name of the *Capture* task if it is running on the same MVS, or in the same JES3 multisystem complex.
- MAXLAG – maximum Synchpoint Lag time (in seconds) to allow; if it is exceeded, generate a warning message. (Note that *Apply* can

have an error, wait for a few minutes, then successfully retry the cycle; and if the MAXLAG time is longer than *Apply*'s ERRWAIT time, no warning may be generated. This could be what you want.)

- WAIT – wait time between monitor snapshots of the control tables in minutes (1-99) or as minutes/seconds ('mmss'). This should be set longer than twice the *Capture* commit time, longer than two average *Apply* cycles, and at least as long as the MAXLAG time to enable the best problem analysis by the monitor; but if WAIT is shorter, more general warning messages will be generated whenever lag exceeds MAXLAG.
- MSGS – where the monitor should send any warning messages:
  - 'CONSOLE' – only to MVS system console (and syslog)
  - 'TSO' – only to specified TSO userids
  - 'BOTH' – to both console and TSO (this is the default)
  - 'NONE' – no warnings sent (but still written in report).
- NOTIFY – list of TSO users to receive warning messages, when MSGS is set to 'TSO' or 'BOTH'.
- END – time ('hh:mm') for monitor to automatically stop today, or tomorrow if it is already later than that time.

The monitor was developed using MAXLAG times from 20 to 3,600 seconds, and WAIT times from 5 seconds to 20 minutes. It functions OK with any combination of those ranges, and presumably is OK with even longer time parameter values.

INSTALLATION

The following steps are required for installation:

- Add index to ASN.IBMSNAP_APPLYTRAIL
- Create DCLGENS for the ASN tables
- Compile/link DPRMON

- Bind DPRMON
- Assemble/link WAITA
- Copy DPROPMON into library
- Create Monitor started task procedure.

ADD INDEX TO ASN.IBMSNAP_APPLYTRAIL TO IMPROVE MONITOR PERFORMANCE

```
CREATE INDEX ASN.IBMSNAP_APPLYTRAIX
       ON ASN.IBMSNAP_APPLYTRAIL
     ( LASTRUN              ASC  )
       USING STOGROUP sto-group
             PRIQTY    pppp
             SECQTY    ssss
       BUFFERPOOL BPx
```

CREATE DCLGENs FOR THE ASN TABLES

Use IBM's DB2I dialog to create DCLGENs in PL/I format with options:

```
COLUMN LABEL .... ===> NO
STRUCTURE NAME .. ===>
FIELD NAME PREFIX ===>
DELIMIT DBCS .... ===> YES
COLUMN SUFFIX ... ===> NO
INDICATOR VARS .. ===> NO

 Name            Table
-------    --------------------
DCLREGR    ASN.IBMSNAP_REGISTER
DCLSUBS    ASN.IBMSNAP_SUBS_SET
DCLCCPP    ASN.IBMSNAP_CCPPARMS
DCLPRUN    ASN.IBMSNAP_PRUNCNTL
```

Put these members in the same source library as the DPRMON program.

COMPILE/LINK PL/I PROGRAM DPRMON

This has been tested using PLI Version 2.3 and VisualAge PL/I. Use standard DSNHPLI procedure for Version 2 of PL/I. For VisualAge

PL/I you may need to create your own JCL based on DSNHPLI, but with changes to the compile steps plus an added pre-link step, as in the procedure IBMZCPL.

```
/*             DATAPROPAGATOR BATCH MONITOR                          */
DPRMON: PROC(PARMS) OPTIONS(MAIN) ORDER;
/*********************************************************************
*  <<<<<<<<<<<<<< MONITOR FOR DB2 VERSION 6 (OR LATER) >>>>>>>>>>>>>  *
*                                                                    *
*  THIS PROGRAM IS INVOKED IN BATCH BY REXX EXEC: DPROPMON, TO QUERY *
*  VARIOUS DB2 TABLES AND RETRIEVE INFORMATION ABOUT THE STATUS OF   *
*  A DATAPROPAGATOR APPLY SUBSCRIPTION SET, SPECIFIED IN THE INPUT   *
*  PARAMETER.  DPROPMON EXEC INVOKES THIS PROGRAM EVERY 5 MINUTES    *
*  (DEFAULT) AND WRITES A REPORT.                                    *
*                                                                    *
*  TABLE ASN.IBMSNAP.SUBS_SET (ON THE DPROP CONTROL DB2 SUB-SYSTEM)  *
*  IS CHECKED TO MEASURE THE PROPAGATION APPLY LAG                   *
*                                                                    *
*  TABLE ASN.IBMSNAP.APPLYTRAIL (ON DPROP CONTROL DB2 SUB-SYSTEM)    *
*  IS QUERIED TO COUNT PROPAGATION APPLY CYCLES & CHANGES            *
*                                                                    *
*  TABLE ASN.IBMSNAP.REGISTER (ON THE SOURCE DB2 SUB-SYSTEM) IS READ *
*  TO MEASURE THE PROPAGATION CAPTURE LAG                            *
*                                                                    *
*  TABLE ASN.IBMSNAP.CCPPARMS (ON THE SOURCE DB2 SUB-SYSTEM) IS READ *
*  TO GET THE COMMIT INTERVAL FOR CAPTURE                            *
*                                                                    *
*  TABLE ASN.IBMSNAP.PRUNCNTL (ON THE SOURCE DB2 SUB-SYSTEM) IS READ *
*  TO GET LOCATION OF THE DPROP CONTROL SERVER                       *
*                                                                    *
*  INPUT PARAMETERS: PREVIOUS-MONITOR-CYCLE-TIMESTAMP                *
*                    MONITOR CYCLE TIME ('MMSS')                     *
*                    SUBSCRIPTION SET NAME                           *
*                                                                    *
*  EXTERNAL PROGRAM: DSNTIAR (GENERATES FORMATTED ERROR MESSAGES)    *
*                                                                    *
*  IT WRITES THE COLLECTED DATA TO THE DPRWORK FILE, SO THAT THE     *
*  DPROPMON EXEC CAN READ THE VALUES AND WRITE THEM INTO ITS REPORT. *
*                                                                    *
*  WRITTEN BY RON BROWN        VERSION 1.13          AUGUST 2002     *
*********************************************************************/
   DCL (ADDR,DATETIME,NULL,SUBSTR)             BUILTIN;
   DCL PARMS            CHAR(69)   VARYING,     /* INPUT PARAMETERS */
       PREV_MON         CHAR(26)   INIT('?????'),
       WAIT_MIN         PIC'99',
       WAIT_SEC         PIC'99',
       QUALSET          CHAR(37)   VARYING,
       QUAL             CHAR(18)   INIT('???? '),
```

```
        SUBS_SET         CHAR(18)   INIT('???? ');
   DCL CAPT_LOCATION     CHAR(16)   INIT('??????? '),
        TARG_LOCATION    CHAR(16)   INIT('??????? '),
        CNTL_LOCATION    CHAR(16)   INIT('??????? '),
        CURRENT_SERVER   CHAR(18)   INIT(' '),
        LOCATION         CHAR(16)   INIT('LOCAL_DB2 '),
        CNTL_LOC         CHAR(1)    INIT('L'),
        CAPT_LOC         CHAR(1)    INIT('L'),
        TYPE             CHAR(7)    VARYING INIT(''),
        W_MIN            FIXED BIN(31),
        W_SEC            FIXED BIN(31),
        I                FIXED BIN(31),
        SET_NAME         CHAR(8)   INIT('????'),
        SYNCH_DATE       CHAR(1Ø)  INIT('??.??.????'),
        SYNCH_TIME       CHAR(8)   INIT('??.??.??'),
        CURRENT_TIME     CHAR(8)   INIT('??.??.??'),
        LOCAL_TIMESTAMP  CHAR(26)  INIT(' '),
        CUR_MIN5_STAMP   CHAR(26)  INIT(' '),
        LAST_TIMESTAMP   CHAR(26)  INIT('??????????????????????????'),
        LTS_IND          FIXED BIN(15,Ø) INIT(Ø), /* NULL INDICATOR */
        CYCLES           FIXED BIN(31,Ø) INIT(Ø),
        CHANGES          FIXED BIN(31,Ø) INIT(Ø),
        CHANGES_IND      FIXED BIN(15,Ø) INIT(Ø), /* NULL INDICATOR */
        TIME_DIFFERENCE  FIXED BIN(31,Ø),
        CAPTURE_LAG_SECS FIXED BIN(31,Ø),
        SYNCH_LAG_SECS   FIXED BIN(31,Ø),
        APPLY_LAG_SECS   FIXED BIN(31,Ø),
        CYCLES_PIC       PIC'ZZZZZZ9' INIT('      Ø'),
        CHANGES_PIC      PIC'ZZZZZZ9' INIT('      Ø'),
        CAPTURE_LAG_PIC  PIC'ZZZZZZ9' INIT('  99999'),
        COMMIT_INT_PIC   PIC'ZZZZ9'   INIT(' 9999'),
        PRUNE_INT_PIC    PIC'ZZZZZZ9' INIT(' 999999'),
        ACTIVATE_PIC     PIC'Z9'      INIT(' 9'),
        SYNCH_LAG_PIC    PIC'ZZZZZZ9' INIT('  99999'),
        APPLY_LAG_PIC    PIC'ZZZZZZ9' INIT('  99999'),
        SLEEP_MIN_PIC    PIC'ZZZ9'    INIT(' 999'),
        MAX_S_MIN_PIC    PIC'ZZZ9'    INIT(' 999'),
        SQLCODE_C_PIC    PIC'S99999'    INIT('-99999'),
        SQLCODE_A_PIC    PIC'S99999'    INIT('-99999');
/*********************************************************************
*    DECLARATIONS SUPPLIED FROM DB2                                  *
*********************************************************************/
   EXEC SQL INCLUDE SQLCA;         /* DEFINE THE SQLCA              */
   EXEC SQL INCLUDE DCLREGR;       /* DCLGEN FOR IBMSNAP_REGISTER   */
   EXEC SQL INCLUDE DCLSUBS;       /* DCLGEN FOR IBMSNAP_SUBS_SET   */
   EXEC SQL INCLUDE DCLCCPP;       /* DCLGEN FOR IBMSNAP_CCPPARMS   */
   EXEC SQL INCLUDE DCLPRUN;       /* DCLGEN FOR IBMSNAP_PRUNCNTL   */
            /* NOTE: WE DO NOT NEED A DCLGEN FOR IBMSNAP.APPLYTRAIL */
/*********************************************************************
*    DECLARATIONS FOR ASSEMBLER PROGRAM DSNTIAR AND ALL OTHER OUTPUT *
```

```
*******************************************************************/
   DCL DSNTIAR          ENTRY OPTIONS(ASM,INTER,RETCODE),
       DATA_LEN         FIXED BIN(31) INIT(132),    /* OUTPUT LRECL */
       DATA_DIM         FIXED BIN(31) INIT(10),
       ERROR_C_HEADING  CHAR(DATA_LEN) INIT(' '),
       1 ERROR_C_MESSAGE  AUTOMATIC,
         3 ERROR_C_LEN  FIXED BIN(15) UNAL INIT((DATA_LEN*DATA_DIM)),
         3 ERROR_C_TEXT(DATA_DIM) CHAR(DATA_LEN) INIT(''),
       ERROR_A_HEADING  CHAR(DATA_LEN) INIT(' '),
       1 ERROR_A_MESSAGE  AUTOMATIC,
         3 ERROR_A_LEN  FIXED BIN(15) UNAL INIT((DATA_LEN*DATA_DIM)),
         3 ERROR_A_TEXT(DATA_DIM) CHAR(DATA_LEN) INIT(''),
       OUTLIST          CHAR(DATA_LEN)  INIT(' '),
       NULL_LINE        CHAR(DATA_LEN)  INIT(''),
       DPRWORK          FILE RECORD OUTPUT;   /* REPORT OUTPUT FILE */
1/*****************************************************************
 *    THIS IS WHERE THE ACTION BEGINS!                             *
 *****************************************************************/
   CALL INPUT_PARMS;
   /*-----------------------------------------------------------*/
   /*  ASSUME THAT WE ARE ON THE DPROP CONTROL SSID,            */
   /*  QUERY SUBS_SET & APPLYTRAIL TABLES TO GET DATA           */
   /*-----------------------------------------------------------*/
   CALL QUERY_APPL;
   /*-----------------------------------------------------------*/
   /*  IF WE ARE ON THE CONTROL SSID, WE MUST CONNECT TO THE */
   /*  SOURCE SSID TO GET THE CAPTURE DATA.                     */
   /*  (USING CAPT_LOCATION FOUND IN SUBS_SET TABLE)            */
   /*-----------------------------------------------------------*/
   IF SQLCODE = 0 & CAPT_LOCATION ¬= CNTL_LOCATION THEN
      CALL CONNECT_REMOTE ('CAPTURE',CAPT_LOCATION);
   /*-----------------------------------------------------------*/
   /*  WE ARE EITHER:                                           */
   /*     A) STARTING ON THE CONTROL SSID AND HAVE NOW          */
   /*        CONNECTED SUCCESSFULLY TO THE SOURCE SSID.         */
   /*     B) NOT ON THE CONTROL SSID - SO WE'LL ASSUME WE       */
   /*        ARE ON THE SOURCE SSID.                           */
   /*  QUERY TABLES TO GET CAPTURE LAG, COMMIT INTERVAL &       */
   /*     LOCATION OF DPROP SERVERS                             */
   /*-----------------------------------------------------------*/
   IF SQLCODE = 0 | SQLCODE = 100 | SQLCODE = -204 THEN
      CALL QUERY_CAPT;              /* -204 = UNDEFINED NAME */
   /*-----------------------------------------------------------*/
   /*  IF WE STARTED ON THE SOURCE SSID AND THE CONTROL SSID */
   /*  IS REMOTE, CONNECT TO THAT CONTROL SSID AND GET THE      */
   /*  APPLY LAG & CYCLE DATA.                                  */
   /*  (USING CNTL_LOCATION FOUND IN PRUNCNTL TABLE)            */
   /*-----------------------------------------------------------*/
   IF SQLCODE_A_PIC ¬= '+00000' & SQLCODE_C_PIC = '+00000'
    & CAPT_LOCATION ¬= CNTL_LOCATION THEN DO;
```

```
      CALL CONNECT_REMOTE ('CONTROL',CNTL_LOCATION);
      IF SQLCODE = Ø THEN
         CALL QUERY_APPL;     /* TRY THIS AGAIN */
      END;
   /* 'EXEC SQL RELEASE ALL' & 'EXEC SQL COMMIT' ARE NOT NECESSARY */
   /*-------------------------------*/
   /* ADJUST LAG TIMES (IF NECESSARY) */
   /*-------------------------------*/
   IF SQLCODE_A_PIC = '+ØØØØØ' THEN          /* WE HAVE LAG TIMES   */
      CALL ADJUST_LAG_TIMES;
   /*-----------------------------------------------------*/
   /*  WRITE MESSAGE(S) INTO FILE WITH DDNAME=DPRWORK      */
   /*-----------------------------------------------------*/
   OPEN FILE(DPRWORK);
   CALL WRITE_DPRWORK;
   CLOSE FILE(DPRWORK);
   /*--------*/
   /*  EXIT  */
   /*--------*/
   RETURN;
1/*===================================================================*/
 /* GET INPUT PARAMETERS                                             */
 /*-------------------------------------------------------------------*/
 INPUT_PARMS: PROC;
    PREV_MON = SUBSTR(PARMS,1,26);   /* LAST MONITOR CYCLE TIMESTAMP */
    WAIT_MIN = SUBSTR(PARMS,28,2);   /* MONITOR CYCLE TIME 'MM..'    */
    WAIT_SEC = SUBSTR(PARMS,3Ø,2);   /* MONITOR CYCLE TIME '..SS'    */
    QUALSET  = SUBSTR(PARMS,33);     /* REST OF INPUT PARMS          */
    I = INDEX(QUALSET,' ');
    QUAL = SUBSTR(QUALSET,1,I-1);    /* NAME OF APPLY QUALIFIER      */
    SUBS_SET = SUBSTR(QUALSET,I+1);  /* NAME OF SUBSCRIPTION SET     */
    W_MIN = WAIT_MIN;                /* CONVERT PIC -> FIXED BIN     */
    W_SEC = WAIT_SEC;
    END INPUT_PARMS;
 /*===================================================================*/
 /* QUERY SUBS_SET & APPLYTRAIL TABLES TO GET APPLY DATA              */
 /*-------------------------------------------------------------------*/
 QUERY_APPL: PROC;
               /* GET APPLY LAG & VARIOUS PROPAGATION CONTROL INFO */
    EXEC SQL
      SELECT
         ACTIVATE
       , DATE (SYNCHTIME)
       , TIME (SYNCHTIME)
       , TIME (CURRENT_TIMESTAMP)
       ,( DAY    (CURRENT_TIMESTAMP - SYNCHTIME) * 864ØØ
        + HOUR   (CURRENT_TIMESTAMP - SYNCHTIME) * 36ØØ
        + MINUTE (CURRENT_TIMESTAMP - SYNCHTIME) * 6Ø
        + SECOND (CURRENT_TIMESTAMP - SYNCHTIME))
       ,( DAY    (LASTRUN - SYNCHTIME) * 864ØØ
```

```
     + HOUR   (LASTRUN - SYNCHTIME) * 3600
     + MINUTE (LASTRUN - SYNCHTIME) * 60
     + SECOND (LASTRUN - SYNCHTIME))
    , SLEEP_MINUTES
    , MAX_SYNCH_MINUTES
    , SET_NAME
    , SOURCE_SERVER
    , TARGET_SERVER
    , CURRENT SERVER
  INTO :ACTIVATE, :SYNCH_DATE, :SYNCH_TIME, :CURRENT_TIME,
       :SYNCH_LAG_SECS, :APPLY_LAG_SECS, :SLEEP_MINUTES,
       :MAX_SYNCH_MINUTES, :SET_NAME,
       :DCLIBMSNAP_SUBS_SET.SOURCE_SERVER,
       :DCLIBMSNAP_SUBS_SET.TARGET_SERVER,
       :CURRENT_SERVER
  FROM ASN.IBMSNAP_SUBS_SET
  WHERE SYNCHTIME < LASTRUN
        AND REFRESH_TIMING IN ('R', 'B')
        AND UPPER(APPLY_QUAL) = :QUAL
        AND UPPER(SET_NAME) = :SUBS_SET
  WITH UR ;
IF SQLCODE ¬= 0 THEN DO;
   ERROR_A_HEADING = ' CONTROL AT '|| LOCATION ||
                 ' SELECT ... FROM ASN.IBMSNAP_SUBS_SET';
   IF SQLCODE = 100 THEN
            LAST_TIMESTAMP = PREV_MON;
   END;
ELSE DO;
   ACTIVATE_PIC  = ACTIVATE;
   SLEEP_MIN_PIC = SLEEP_MINUTES;
   MAX_S_MIN_PIC = MAX_SYNCH_MINUTES;
   CNTL_LOCATION = SUBSTR(CURRENT_SERVER,1,16);
   CAPT_LOCATION = SUBSTR(DCLIBMSNAP_SUBS_SET.SOURCE_SERVER,1,16);
   TARG_LOCATION = SUBSTR(DCLIBMSNAP_SUBS_SET.TARGET_SERVER,1,16);
  /*-------------------------------------------------*/
  /* GET NUMBER OF SUCCESSFUL APPLY CYCLES & CHANGES  */
  /* + THE LAST LASTRUN VALUE - TO BE NEXT :PREV_MON  */
  /*-------------------------------------------------*/
   EXEC SQL
     SELECT
       CURRENT_TIMESTAMP - 5 SECONDS,
       COUNT(*),
       SUM(SET_INSERTED+SET_DELETED+SET_UPDATED+SET_REWORKED),
       MAX(LASTRUN)
     INTO :CUR_MIN5_STAMP, :CYCLES, :CHANGES:CHANGES_IND,
          :LAST_TIMESTAMP:LTS_IND
     FROM ASN.IBMSNAP_APPLYTRAIL
     WHERE
       (CASE WHEN :PREV_MON = '??????????????????????????'
          THEN CURRENT_TIMESTAMP - :W_MIN MINUTES - :W_SEC SECOND
```

```
             ELSE :PREV_MON
             END) < LASTRUN  AND
           STATUS >= Ø       AND
           UPPER(APPLY_QUAL) = :QUAL  AND
           UPPER(SET_NAME) = :SUBS_SET
        WITH UR;
      IF SQLCODE = Ø | SQLCODE = 1ØØ | SQLCODE = -181 THEN DO;
         IF CHANGES_IND < Ø THEN        /* -181 = INVALID DATETIME */
            CHANGES = Ø;
         IF LTS_IND < Ø THEN DO;   /* SET VALID LAST LASTRUN VALUE */
            IF PREV_MON = '????????????????????????' THEN
               LAST_TIMESTAMP = CUR_MIN5_STAMP;    /* TABLE EMPTY? */
            ELSE
               LAST_TIMESTAMP = PREV_MON;
            END;
         IF SQLCODE = 1ØØ THEN           /* Ø ROWS IS NOT AN ERROR */
            SQLCODE = Ø;
         END;
      IF SQLCODE ¬= Ø THEN
         ERROR_A_HEADING = ' CONTROL AT '|| LOCATION ||
                    ' SELECT ... FROM ASN.IBMSNAP_APPLYTRAIL';
      END;
   IF SQLCODE ¬= Ø THEN
      CALL DSNTIAR (SQLCA, ERROR_A_MESSAGE, DATA_LEN);
   ELSE DO;
      CYCLES_PIC    = CYCLES;        /* CONVERT FIXED BIN -> CHAR */
      CHANGES_PIC   = CHANGES;
      END;
   SQLCODE_A_PIC = SQLCODE;
   END QUERY_APPL;
/*====================================================================*/
/* QUERY CAPTURE TABLES TO GET CAPTURE LAG, COMMIT INTERVAL,       */
/*     PRUNE INTERVAL AND LOCATION OF CNTL SERVER                  */
/*--------------------------------------------------------------------*/
QUERY_CAPT: PROC;
      /*----------------------------------*/
      /* GET THE CAPTURE LAG              */
      /*----------------------------------*/
      EXEC SQL
        SELECT  HOUR   (CURRENT_TIMESTAMP - SYNCHTIME) * 36ØØ
             +  MINUTE (CURRENT_TIMESTAMP - SYNCHTIME) * 6Ø
             +  SECOND (CURRENT_TIMESTAMP - SYNCHTIME)
        INTO :CAPTURE_LAG_SECS
        FROM ASN.IBMSNAP_REGISTER
           WHERE GLOBAL_RECORD = 'Y'
        WITH UR ;
      IF SQLCODE ¬= Ø THEN
         ERROR_C_HEADING = ' CAPTURE AT '|| LOCATION ||
                  ' SELECT ... FROM ASN.IBMSNAP_REGISTER';
      ELSE DO;
```

```
         /*---------------------------------*/
         /* GET THE CAPTURE COMMIT INTERVAL  */
         /*---------------------------------*/
         EXEC SQL
           SELECT  COMMIT_INTERVAL, PRUNE_INTERVAL
           INTO :COMMIT_INTERVAL, :PRUNE_INTERVAL
           FROM ASN.IBMSNAP_CCPPARMS
           WITH UR ;
         IF SQLCODE ¬= Ø THEN
            ERROR_C_HEADING = ' CAPTURE AT '|| LOCATION ||
                    ' SELECT ... FROM ASN.IBMSNAP_CCPPARMS';
         ELSE DO;
            /*---------------------------------*/
            /* GET LOCATION OF CONTROL SERVER   */
            /*---------------------------------*/
            EXEC SQL
              SELECT  DISTINCT(SET_NAME), CURRENT SERVER,
                      CNTL_SERVER, TARGET_SERVER
              INTO :SET_NAME, :CURRENT_SERVER, :CNTL_SERVER,
                   :DCLIBMSNAP_PRUNCNTL.TARGET_SERVER
              FROM ASN.IBMSNAP_PRUNCNTL
              WHERE UPPER(SET_NAME) = :SUBS_SET AND
                    UPPER(APPLY_QUAL) = :QUAL
              WITH UR ;
            IF SQLCODE ¬= Ø THEN
               ERROR_C_HEADING = ' CAPTURE AT '|| LOCATION ||
                       ' SELECT ... FROM ASN.IBMSNAP_PRUNCNTL';
            END;
         END;
   IF SQLCODE ¬= Ø THEN
      CALL DSNTIAR (SQLCA, ERROR_C_MESSAGE, DATA_LEN);
   ELSE DO;
      COMMIT_INT_PIC = COMMIT_INTERVAL;
      PRUNE_INT_PIC = PRUNE_INTERVAL;
      CAPTURE_LAG_PIC = CAPTURE_LAG_SECS;
      IF CAPT_LOCATION = '??????? ' THEN
         CAPT_LOCATION = SUBSTR(CURRENT_SERVER,1,16);
      CNTL_LOCATION = SUBSTR(CNTL_SERVER,1,16);
      TARG_LOCATION = SUBSTR(DCLIBMSNAP_PRUNCNTL.TARGET_SERVER,1,16);
      END;
   SQLCODE_C_PIC = SQLCODE;
   END QUERY_CAPT;
/*================================================================*/
/* CONNECT TO A REMOTE DB2 SSID                                   */
/*----------------------------------------------------------------*/
CONNECT_REMOTE: PROC (TYPE,LOCATION);
   DCL TYPE     CHAR(7) VARYING,
       LOCATION CHAR(16);
   EXEC SQL
     CONNECT TO :LOCATION;
```

```
   IF TYPE = 'CAPTURE' THEN DO;
      CAPT_LOC = 'R';
      SQLCODE_C_PIC = SQLCODE;
      IF SQLCODE ¬= Ø THEN DO
         ERROR_C_HEADING = ' CAPTURE:  CONNECT TO '|| LOCATION;
         CALL DSNTIAR (SQLCA, ERROR_C_MESSAGE, DATA_LEN);
         END;
      END;
   IF TYPE = 'CONTROL' THEN DO;
      CNTL_LOC = 'R';
      SQLCODE_A_PIC = SQLCODE;
      IF SQLCODE ¬= Ø THEN DO
         ERROR_A_HEADING = ' CONTROL:  CONNECT TO '|| LOCATION;
         CALL DSNTIAR (SQLCA, ERROR_A_MESSAGE, DATA_LEN);
         END;
      END;
   /*-------------------------------------------------------------*/
   /* GET TIME DIFFERENCE BETWEEN LOCAL AND REMOTE DB2 SYSTEMS */
   /*-------------------------------------------------------------*/
   IF SQLCODE = Ø THEN DO;
      LOCAL_TIMESTAMP = SUBSTR(DATETIME,1,4)||'-'        /* YYYY   */
                      ||SUBSTR(DATETIME,5,2)||'-'        /* MM     */
                      ||SUBSTR(DATETIME,7,2)||'-'        /* DD     */
                      ||SUBSTR(DATETIME,9,2)||'.'        /* HH     */
                      ||SUBSTR(DATETIME,11,2)||'.'       /* MM     */
                      ||SUBSTR(DATETIME,13,2)||'.'       /* SS     */
                      ||SUBSTR(DATETIME,15,3)||'5ØØ';    /* ...... */
      EXEC SQL
        SELECT
          SECOND ( CURRENT_TIMESTAMP - :LOCAL_TIMESTAMP )
        INTO :TIME_DIFFERENCE
        FROM SYSIBM.SYSDUMMY1
        WITH UR ;
      END;
   END CONNECT_REMOTE;
/*=============================================================*/
/* CORRECT LAG TIMES TO ALLOW FOR TIME DIFFERENCES BETWEEN DB2S    */
/*-------------------------------------------------------------*/
ADJUST_LAG_TIMES: PROC;
   SELECT;
      WHEN (CAPT_LOC = 'R') DO;
        SYNCH_LAG_PIC = SYNCH_LAG_SECS + TIME_DIFFERENCE;
        APPLY_LAG_PIC = APPLY_LAG_SECS + TIME_DIFFERENCE;
        END;
      WHEN (CNTL_LOC = 'R') DO;
        SYNCH_LAG_PIC = SYNCH_LAG_SECS - TIME_DIFFERENCE;
        APPLY_LAG_PIC = APPLY_LAG_SECS - TIME_DIFFERENCE;
        END;
      OTHERWISE DO;  /* CNTL & CAPT BOTH LOCAL!! */
        SYNCH_LAG_PIC = SYNCH_LAG_SECS;
```

```
         APPLY_LAG_PIC = APPLY_LAG_SECS;
         END;
      END;
   END ADJUST_LAG_TIMES;
/*====================================================================*/
/* WRITE LINES INTO FILE DDNAME=DPRWORK                               */
/* - THE FIRST 2 LINES HAVE SQLCODES & ALL RETRIEVED DATA VALUES.     */
/* - IN CASE OF BAD SQL CODES, THE DB2 ERROR MESSAGES GENERATED       */
/*     BY DSNTIAR ARE WRITTEN INTO THE FILE TOO.                      */
/*--------------------------------------------------------------------*/
WRITE_DPRWORK: PROC;
   /*-----------------------------------------------------*/
   /*  WRITE TWO LINES INTO FILE                          */
   /*-----------------------------------------------------*/
   OUTLIST = SQLCODE_C_PIC ||' '|| CAPT_LOC ||' '||
             SQLCODE_A_PIC ||' '|| CNTL_LOC ||' '||
             CAPT_LOCATION ||' '|| CNTL_LOCATION ||' '||
             TARG_LOCATION ||' '|| LAST_TIMESTAMP ||' '||
             TIME_DIFFERENCE;
   WRITE FILE(DPRWORK) FROM(OUTLIST);
   OUTLIST = CURRENT_TIME || CAPTURE_LAG_PIC ||' '||
             SET_NAME ||' '|| SYNCH_DATE ||' '|| SYNCH_TIME ||
             SYNCH_LAG_PIC || APPLY_LAG_PIC || ACTIVATE_PIC ||
             COMMIT_INT_PIC || SLEEP_MIN_PIC ||' '||
             MAX_S_MIN_PIC ||' '|| PRUNE_INT_PIC ||' '||
             CYCLES_PIC ||' '|| CHANGES_PIC;
   WRITE FILE(DPRWORK) FROM(OUTLIST);
   /*-----------------------------------------------------*/
   /*  IF BAD SQLCODE, WRITE DSNTIAR MESSAGES INTO FILE   */
   /*-----------------------------------------------------*/
   IF SQLCODE_A_PIC ¬= '+00000' THEN DO;
      WRITE FILE(DPRWORK) FROM(ERROR_A_HEADING);    /* ERROR TITLE */
      DO I = 1 TO DATA_DIM;
         IF ERROR_A_TEXT(I) ¬= NULL_LINE THEN
            WRITE FILE(DPRWORK) FROM(ERROR_A_TEXT(I)); /* DB2 MSGS */
         END;
      END;
   IF SQLCODE_C_PIC ¬= '+00000' & SQLCODE_C_PIC ¬= '-99999' THEN DO;
      WRITE FILE(DPRWORK) FROM(ERROR_C_HEADING);
      DO I = 1 TO DATA_DIM;
         IF ERROR_C_TEXT(I) ¬= NULL_LINE THEN
            WRITE FILE(DPRWORK) FROM(ERROR_C_TEXT(I));
         END;
      END;
   END WRITE_DPRWORK;
END DPRMON;
```

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

Ron Brown (Consultant)

- General administration tools – Control Center (*db2cc*), Journal (*db2journal*), Replication Center (*db2rc*), Task Center (*db2tc*).
- Information – Information Center (*db2ic*).
- Monitoring tools – Event Analyzer (*db2eva*), Health Center (*db2hc*), Indoubt Transaction Manager (*db2indbt*), Memory Visualizer (*db2memvis*).
- Set-up tools – Configuration Assistant (*db2ca*), Register Visual Studio Add-Ins (*db2vsregister*), Satellite Synchronizer.

Obviously, some of these are too long to type in every time, so I created shortcuts for things like db2journal. The 'Register Visual Studio Add-Ins' and 'Satellite Synchronize' are not ones that I have investigated.

Some key points to note are that the Stored Procedure Builder has been replaced with the functionally-enhanced Development Center and the Project Deployment Tool. Also, replication has been taken out of the Control Center and given its own GUI. The performance monitor has also been taken out of the Control Center and there is a whole new section on monitoring tools.

The Health Center GUI is a departure from the V7 performance monitor, and seems, at first glance, to be a vast improvement. It allows you to set thresholds for various parameters and then send e-mail or pager messages to a contact list if any threshold is breached. I have not tried this out yet, but will be doing so shortly, so watch this space for a future article on the subject.

NEW FEATURES

Some of the new features that I think deserve special mention include being able to: use multi-dimensional clustering; load data 'on-line'; reorganize a table on-line; specify a different log path name for the secondary logs other than the primary log path name with a '2'; 'quiesce' a database; reorgchk using a schema as a filter; rename a table; write event monitor information to DB2 tables; configure dbm and db parameters on-line (with the get db cfg and get dbm cfg commands being enhanced with a 'show detail' parameter); alter a bufferpool size on-line; and insert into UNIONL ALL views. I will show how each of these features works.

Multi-dimensional clustering

Up to now you were able to cluster your data on disk according to one index only. This is fine if you are retrieving data based on that one index (ie where year = 2002), but what happens if your next query wants to retrieve data based on a different index (ie where region = 'NL')? This is where multi-dimensional clustering comes in. It allows you to build multiple indexes, which DB2 treats as clustered. So in our previous example, we would build a clustering index on the year column and the region column. The syntax for creating multi-dimensional indexes is very easy, as shown below:

```
>db2 create table mdc02 (year int, reg char(5), mach char(5), col
char(5), amtsold int) organize by (year, reg)
```

You can see the indexes that were produced by:

```
>db2 select substr(indname,1,18), substr(colnames,1,10), indextype from
syscat.indexes where tabname = 'MDC02'

1                  2          INDEXTYPE
------------------ ---------- ---------
SQL020804192836660 +REG+YEAR  BLOK
SQL020804192837110 +REG       DIM
SQL020804192837220 +YEAR      DIM
```

You can see that DB2 has created two indexes of type DIM, and one of type BLOK.

You can still create indexes on the other columns in the normal way.

You don't have to specify anything in the SQL that you run to tell it to use the MDC indexes – DB2 will use them if the optimizer thinks they are appropriate. So how can you tell if you are using them or not? Well, if you look at the EXPLAIN output for the query, you will see that it is using index SQL020804192837220, which proves that the optimizer has chosen an MDC index. I certainly wouldn't make every index an MDC index; their introduction needs careful planning, but if implemented correctly, they can certainly improve query performance. More on MDC in a future article.

Load utility improvements

In V8, you can now load a table without locking out the tablespace, or even stopping read access to the table itself. There is a new keyword,

ALLOW READ ACCESS, which allows users to still read the existing data in a table whilst loading rows into the table. It is very difficult to show this in action in an article, but give it a go and see for yourself.

Create table fred01 with 10 rows in it. Create a load file with 10,000 rows in it. From one CLP session start loading fred01 from the load file, then switch to another CLP session and do a **select * from fred01**, which should return the 10 rows.

Reorganizing a table on-line

As with the load utility, there is now a new keyword, ALLOW READ ACCESS, on the REORG command. We have had this ability to do on-line REORGs in the OS/390 world for many years, so the on-line feature is a welcome addition in the mid-range arena.

```
>db2 reorg table mdc02 allow read access
DB20000I The REORG command completed successfully.
```

Again, it is difficult to show it in action in an article.

Create table fred03 with 100,000 rows in it. From one CLP session issue the REORG command with "allow read access", then switch to another CLP session and do a **select * from fred03 where ..**, which should return some rows.

Mirrored log path

It is now possible to mirror the primary logs to a path which is not just the primary log path name with a '2' appended to it! This is shown below (you need to create the new log path directory before issuing the db cfg update command, otherwise you will get an SQL5099N error):

```
>db2 get db cfg for sample | find "LOG"
Path to log files                  = C:\DB2\NODE0000\SQL00002\SQLOGDIR\
Mirror log path   (MIRRORLOGPATH) =

>db2 update db cfg for sample using mirrorlogpath c:\samplog
DB20000I  The UPDATE DATABASE CONFIGURATION command completed
successfully.
SQL1363W One or more of the parameters submitted for immediate
modification were not changed dynamically. For these configuration
parameters, all applications must disconnect from this database before
```

```
the changes become effective.
>db2 get db cfg for sample | find "LOG"
Path to log files                 = C:\DB2\NODE0000\SQL00002\SQLOGDIR\
Mirror log path    MIRRORLOGPATH) = c:\samplog\NODE0000\
```

So if I now disconnect from the database and reconnect, and do a **>cd \samplog\NODE0000**, I see:

```
C:\samplog\NODE0000>dir
Directory of C:\samplog\NODE0000

            24,576 S0000000.LOG
            24,576 S0000001.LOG
            24,576 S0000002.LOG
               512 SQLLPATH.TAG
```

Obviously, this mirrored path can be a filesystem on a different machine at a different site, thus helping in disaster recovery planning.

Quiescing a database

In previous versions of DB2 you could issue a **force application all** command, which would indeed force off all currently-connected applications to the instance, but there was nothing to stop these applications reconnecting a few seconds later! What was required was a command similar to the **start in maint mode** command in DB2 on OS/390. What we have is a quiesce database command. In the following commands, I am connected to database sample from a CLP session:

```
>db2 quiesce database immediate force connections

>db2 get snapshot for database on sample | find "status"
Database status                           = Quiesced

>db2 unquiesce database

>db2 get snapshot for database on sample | find "status"
Database status                           = Active
```

As you can see from the above commands, you can use the **get snapshot** command to see if the database is actually quiesced or not. I didn't need to set any snapshot switches to get the database status. You take the database out of quiesce mode using the unquiesce command, as shown above.

Enhancement to the reorgchk command

What's new with the reorgchk command is that you can now specify a schema name, which is shown below. Be aware, that the default is still UPDATE statistics, which is why I specified current statistics. The command shown will perform a reorgchk on all tables in the db2admin schema:

```
>db2 reorgchk current statistics on schema db2admin
```

Being able to rename a table

We now have the ability to rename a table, rather than creating a new table 'like' the existing one and then selecting into it. I have shown this below, where I first show that table sue doesn't exist, then create table fred03, select from it and finally rename it to sue – and select from sue.

```
>db2 select * from sue
SQL0204N  "DB2ADMIN.SUE" is an undefined name.  SQLSTATE=42704

>db2 create table fred03 (year int)
DB20000I  The SQL command completed successfully.

>db2 insert into fred03 values(2002)
DB20000I  The SQL command completed successfully.

>db2 rename table fred03 to sue
DB20000I  The SQL command completed successfully.

>db2 select * from sue

YEAR
-----------
       2002
```

Just to prove that table fred03 no longer exists, do a:

```
>db2 select * from fred03
SQL0204N  "DB2ADMIN.FRED03" is an undefined name.  SQLSTATE=42704
```

There are, however, restrictions as to which tables you can and cannot rename. For example, if a table has a view created on it, then you cannot rename the table, as shown below (following on from the above example):

```
>db2 create view harry as select * from sue
DB20000I  The SQL command completed successfully.
```

```
>db2 rename table sue to mary

DB21034E The command was processed as an SQL statement because it was
not a valid Command Line Processor command. During SQL processing it
returned:
SQL0750N The source table cannot be renamed because it is referenced in
a view, materialized query table, trigger, SQL function, SQL method,
check constraint, or referential constraint. SQLSTATE=42986
```

You can see that the rename failed, because I had a view harry on table sue. The other restrictions are detailed in the SQL reference manual. You can also rename an index, although I found this less useful than being able to rename a table!

Writing event monitor output to a table

In the past you could write event monitor output only to a file or a pipe, and then post-process the file etc. A new option is to write the event monitor output to DB2 tables. The command reference manual says you can use the db2evtbl command (db2evtbl -schema smith -evm foo database, tables, tablespaces, bufferpools) to generate the DDL to set up an event monitor to write to a table – however, I could not get this command to work, so I created the monitor manually, as:

```
>db2 create event monitor mon1 for statements write to table
```

This creates the following tables in the default tablespace (the SQL reference manual tells you which tables are created for each monitor setting. I don't create a SUBSECTION table because I am running this on an EE system):

```
CONNHEADER_MON1, CONTROL_MON1, STMT_MON1
```

Activate the monitor using:

```
>db2 set event monitor mon1 state = 1
```

Check that the monitor is active:

```
>db2 SELECT evmonname ,EVENT_MON_STATE(evmonname) AS state , io_mode
FROM syscat.eventmonitors
```

Run some SQL (eg select count(*) from syscat.tables).

You then have to turn the event monitor off before it writes the information to the tables:

```
>db2 set event monitor mon1 state = 0
```

Now you can look at the SQL that has been run using:

```
>db2 select substr(stmt_text,1,50) from stmt_mon1
```

Note, that the stmt_text column is defined as a CLOB. The *SQL Reference* manual details all the columns in the stmt_mon1 table. You can create your own table with just the columns you want. You can also specify a tablespace for the monitor tables. I have just scratched the surface as to what you can do with these tables, but have, hopefully, shown how easy they are to use.

Configurable on-line parameters

There has long been a desire to be able to update database manager and database configuration parameters without the need to either stop/start the instance or disconnect all users from the database.

We now have the option either to do the update 'on-line' and pick up the change immediately, or to defer the implementation until the next instance stop/start or database reconnection. There is a new keyword with the get dbm cfg and get db cfg commands, which shows you what the current value of any particular parameter is (current value), and what it will be at the next invocation (delayed value). This keyword is **show detail**, and below is an example. There is also a new keyword on the update db cfg command, which is either deferred or immediate (note that immediate is the default!). This allows you to update some parameters immediately.

The example below shows an update to a db cfg variable (BUFFPAGE), which cannot be done immediately because it needs all applications to disconnect from the database before it can take effect:

```
>db2 get db cfg for sample show detail | find "BUFFPAGE"
Buffer pool size (pages)  (BUFFPAGE) = 250                    250
```

The first value is the current setting, and the second value is what it will be set to at the next invocation. Now, suppose we update BUFFPAGE to 500:

```
>db2 update db cfg for sample using BUFFPAGE 500
DB20000I The UPDATE DATABASE CONFIGURATION command completed
successfully.
SQL1363W One or more of the parameters submitted for immediate
modification were not changed dynamically. For these configuration
```

```
parameters, all applications must disconnect from this database before
the changes become effective.
```

Now if we issue the get db cfg command again we can see that the delayed value is 500, so this is the value that will be used when all applications disconnect from the database:

```
>db2 get db cfg for sample show detail | find "BUFFPAGE"
Buffer pool size (pages)  (BUFFPAGE) = 250                     500
```

The SQL reference manual gives details of which variables can be immediately updated and which ones cannot.

Changing bufferpool sizes on-line

I guess the requirement to change the bufferpool sizes on-line followed on from the requirement to change db/dbm cfg parameters on-line. You can now change bufferpool sizes on-line. I did this as follows:

```
>db2 select substr(bpname,1,15), npages from syscat.bufferpools

1               NPAGES
--------------- -----------
IBMDEFAULTBP            250

>db2 alter bufferpool ibmdefaultbp size 100000
DB20000I  The SQL command completed successfully.

>db2 select substr(bpname,1,15), npages from syscat.bufferpools

1               NPAGES
--------------- -----------
IBMDEFAULTBP         100000
```

If you are on a Windows system, you can use the task manager to see that the amount of memory being used has indeed increased after the **alter** command has been issued.

If you try to over-allocate the memory, you will get a message like:

```
>db2 alter bufferpool ibmdefaultbp size 500000

SQL20189W The buffer pool operation (CREATE/ALTER) will not take effect
until the next database startup due to insufficient memory.
SQLSTATE=01657
```

You can also use the **alter** command to reduce a bufferpool size. I would try to reduce the value gradually, rather than in one go.

Being able to insert into UNION ALL views

We have long been able to select from UNION ALL views, but not insert. This has now changed, as is shown below:

```
>db2 create table mary1 (id int check (id between  0 and  9))

>db2 create table mary2 (id int check (id between 10 and 19))

>db2 create table mary3 (id int check (id between 20 and 29))

>db2 create view mall as select * from mary1 union all select * from
mary2 union all select * from mary3

>db2 insert into mall values(9)
DB20000I  The SQL command completed successfully.

>db2 select * from mary1

ID
-----------
          9

>db2 select * from mary2

ID
-----------

  0 record(s) selected.
```

As you can see, a row has successfully been inserted into table mary1. For this to work, you must use the **check** option on the column at table create time, otherwise you will get an error – shown below – which is not unexpected because DB2 doesn't know where to put the inserted row!

```
SQL20154N The requested insert operation into view "DB2ADMIN.MALL" is
not allowed because no target table can be determined for a row. Reason
code = "2". SQLSTATE=23513
```

I think there is enough in this new version to make us realize that it is not just a cosmetic upgrade, but that it offers substantially improved functionality over V7. I have just picked out the new features that I thought were worth mentioning – see the *What's New* manual for a more exhaustive list of what V8 has to offer, and give it a go!

C Leonard
Freelance Consultant (UK)

Operational Data Store (ODS)

IBM Global Services announced ODS in May 2000 (http://www-4.ibm.com/BI) for real-time access. Gartner Group (http://www4.gartner.com) has included ODS as a component in its Zero Latency Enterprise (ZLE) for near real-time integration of operational and information systems for data and business processes. ZLE provides instantaneous access to enterprise data for immediate business decisions.

ODS

ODS was developed to address the business requirement of making quick assessments. ODS is an architectural construct containing subject-oriented, integrated, volatile, current (or almost current), and detailed operational data. Two new terms need defining:

- Frequency – ODS update rate.
- Velocity – elapsed time between legacy update and its storage in ODS.

Frequency must be sufficient to handle velocity.

ODS BENEFITS

ODS benefits include:

- Improved accessibility to critical operational data.
- Imparting better understanding of customers by making available complete financial metrics and transactions permitting well-informed decisions or providing real-time access to product and service usage data. (Note: *or* is used instead of *and* to signify that there will be multiple ODSs just as there are multiple data marts.)
- Supplying operational reports faster than legacy applications.
- Integrating new and existing applications.
- Shortening DW (data warehouse) population time, because it contains integrated data.

ODS:DW

Critical differences between an ODS and a DW are shown in Figure 1.

ODS	*DW*
High-quality data at detailed level with assured availability.	Data can be imperfect as long as it is sufficient for strategic analysis; does not need high availability.
Contains real-time or near real-time data.	Contains historical data.
Real-time or near real-time data loads.	Batch data loads.
Updated at data field level.	Data is appended; never updated.
Detailed data only.	Summarized and detailed data.
Uses 3NF for rapid data updates.	Usually uses star or dimensional model for better performance.
Similar to OLTP.	Queries can produce large data volumes.
Used for detailed decision-making and operational reporting.	Used for long-term decision-making and management reporting.
Tactical; used at the operational level.	Strategic; used at the management level.
Uses record level access.	Uses result set access.

Figure 1: Critical differences between an ODS and a DW

The most critical difference is that ODS does SQL INSERT, UPDATE, DELETE whereas DW uses *only* APPEND. This dissimilarity means organizations will need both an ODS and a DW if they want real-time detailed data access and strategic analysis. The other important distinction is that an ODS needs high frequency and velocity.

ODS WITHIN BI ARCHITECTURE

ODS and DW are populated from operational and external sources via integration and transformation programs. Two streams are necessary to satisfy their different information needs and to provide *single source population*. ODS can push data to the DW using normal batch methods. It is possible for some data to flow back from the DW to the ODS or from the ODS to operational data sources.

BUSINESS REQUIREMENT EXAMPLES

Business requirements include:

- Banking – provides complete customer credit picture so that an instant increase can be granted.
- Insurance – displays up-to-date view of customer insurance products to Customer Relations Officers.
- Retail – allows suppliers to co-manage inventory.
- Telecommunication – determines which credit cards are used for fraudulent calls.

ODS TYPES

ODS has three architectures although some ODSs have a hybrid architecture. They are:

- Type A – real-time store-and-forward, or batch from data sources.
- Type B – real-time store-and-forward, or batch from data sources; triggers to apply update to operational data.
- Type C – real-time update and access. Fully integrated.

ODS characteristics are shown in Figure 2.

All types have the following common characteristics.

- ODS can be directly updated by front-end applications.
- ODS can be data source for DW using batch processes.

Characteristics	*Type A*	*Type B*	*Type C*
Data source	Operational and other	Operational and other	Operational and other
Real-time ODS update	Yes	Yes	Yes
Direct access from front-end apps	Read/write (usually more reads)	Read/write	Read/write
Data flow to ODS	One way	Both ways	Both ways
Update timing for operational apps	N/A	Asynchronous triggers	Real-time

Figure 2: ODS characteristics

- ODS complements or extends operational applications; it does not replace them.

SELECTING ODS TYPE

Is real time access to transactional date required?

No – use DW or Data Mart.

Yes – use ODS.

Are updates to the data sources volatile?

No – use Type A.

Yes – use Type A or Type B.

Does the solution require a single version of the truth?

No – use Type B.

Yes – use Type C.

DATA WAREHOUSE MANAGER (DB2 WM)

DB2 WM provides a distributed, heterogeneous infrastructure for designing, building, maintaining, governing, and accessing highly-scalable and robust DB2 DWs and ODSs. Data can be moved directly from source to target using DW agents to place transformations on different servers. DW and ODS deployment can be enhanced by using pre-built cleansing and statistical transformations. Capabilities can be expanded with user-defined transformation functions or by using vendors like ETI*EXTRACT or Vality Integrity. IBM products include:

- MQSeries:
  - Works on various platforms.
  - Does codepage translation.
  - Assures delivery.
  - MQSI:
    - Does data integration
    - Does data transformation
    - Does data cleansing
    - Provides intelligent routing.
- DataJoiner:
  - Accesses heterogeneous stores.
  - Accesses federated ODSs.
  - Optimizes database access to Informix, Microsoft, Oracle, Sybase, and Teradata.
- Vality Integrity:
  - Does data cleansing.
  - Does fuzzy matching.
  - Does value and metadata discovery.
- DB2 WM:

  - Does extract.
  - Does transform.
  - Does move.
  - Does load.
  - Provides automation routines.
  - Provides monitoring.
  - Does query and reporting.
  - Provides and/or captures metadata.
  - Can access DB2 family (there are restrictions on DB2 UDB EEE), flat files, IMS, Informix, Oracle, SQL Server, Sybase, and VSAM.
- Data Replication:
  - Updates ODS.
  - Provides transformation flexibility.
  - Optimizes network.
  - Rated as best tool for capturing DB2 and IMS changes.
- ETI*EXTRACT:
  - Can access many legacy systems based on ADABAS, IDMS, and SAP.
  - Can generate COBOL and C code.
  - Does complex transformations.
- Ascential Data Stage:
  - Accesses relational and flat file sources.
  - 350 predefined transformations.
  - 390 COBOL code generations for DB2, VSAM, and flat files.
- Evoke Software:

  - Does data cleansing.
  - Does data profiling.
  - Discovers data relationships.

There are many other vendor tools to provide support.

DATA WAREHOUSE CENTER (DWC)

Integrating DWC management capability with DB2 replication technology provides a potent solution for designing and implementing an ODS. A possible scenario is:

- Use DB2 *Capture* to acquire changed data at the source DB2.
- Use DB2 *Apply* to acquire updates to target tables.
- Use DWC to manage the population process:
  - Single replication process
  - Automate the *Apply* process at the target database.
- Use DB2 Control Center to define the *Capture* process.
- Use DWC to define the replication step.
- Use DWC scheduling function to run *Apply.*
- Use DWC to transform, integrate, and combine the data with other data before updating the ODS.

DWC replication scenario:

- Import metadata about source tables that are registered as replication sources.
- Select replication step and connect input source table.
- Fill in parameters.
- Connect replication step to output table or have DWC generate an output table based on the replication step.
- Define step scheduling after determining whether it is time-based or dependency-based.

DB2 UDB FOR Z/OS CROSS LOADER

IBM has added the Cross Loader function (DB2 CL) to DB2 UDB Load utility. CL loads DB2 for z/OS tables from DB2 UDB tables on distributed platforms such as NT, Unix, or another DB2 for z/OS system. CL combines IBM LOAD utility efficiency with DRDA robustness and flexible SQL. CL enables any SQL SELECT output to be directly loaded into a table on DB2 UDB for z/OS.

The SQL SELECT can access any DRDA server, so the data source can be any DB2 UDB family member or any vendor product that has implemented DRDA. It provides an additional technique for loading small quantities of data from DB2 distributed platforms or another DB2 for z/OS into the ODS.

DB2 CL is a simpler process than unloading data, transferring the unloaded file to the target site, and then running the LOAD utility.

SUMMARY

- ODS contains subject-oriented, integrated, volatile, current, and detailed operational data that allows instant access for rapid business decisions.
- ODS can provide reports more quickly than legacy applications and shorten DW population time.
- Critical differences between ODS and DW are that ODS uses standard SQL with high frequency and velocity, whereas DW uses only APPEND and batch updates.
- ODS and DW are populated using a single source.
- ODS has three architecture types:
  - Type A – real time or batch from data sources
  - Type B – real time or batch with triggers
  - Type C – fully integrated with real-time update and access.
- DB2 WM is the best tool for designing, building, maintaining, governing, and accessing DB2 DWs and ODSs. Products include MQSeries, DataJoiner, Vality Integrity, DB2 WM, Data Replication,

ETI*EXTRACT, Ascential Data Stage, and Evoke Software.

- DWC is the best tool for designing and implementing ODS.
- DB2 UDB for z/OS CROSS LOADER is easier to use than LOAD.

CONCLUSION

Any organization with a successful DW should consider implementing an ODS to provide operational users with real-time data access to let them make decisions quickly.

Eric Garrigue Vesely
Principal/Analyst
Workbench Consulting (Malaysia)

November 1999 – October 2002 index

Items below are references to articles that have appeared in *DB2 Update* since November 1999. References show the issue number followed by the page number(s). Subscribers can download copies of all issues in Acrobat PDF format from Xephon's Web site.

Circle has just launched a new product called VS2, an Enterprise Application Integration tool allowing large companies to transparently convert their legacy VSAM data into DB2 tables

VS2 allows companies to migrate applications from VSAM to DB2 without code changes. The data is in DB2, but the programs can still call VSAM.

VS2 supports extensive re-engineering of the datasets during migration; a VSAM record can be mapped to several DB2 tables, and fields can be converted to more powerful DB2 data types.

The VS2 run-time facility intercepts VSAM calls and directs the request to DB2 or VSAM - depending on whether the data has been migrated or not.

For further information contact:
Circle Computer Group, Queensberry House, Queens Road, Brighton BN1 3XF, UK.
Tel: (01273) 721123.
URL: http://www.circle-group.com/software/softwareintro.htm.

* * *

IBM has announced an open beta version of DB2 Version 8 with new self-management capabilities and broader support for open standards.

New bits include autonomic computing capabilities, including self-managing and self-tuning features, support for Federated Web Services, and claimed greater performance.

The software can now, through the new Health Center, update database administrators on system performance, give advice on problems occurring in the database or application it supports, and alert them that a fix has been generated.

It alerts administrators via e-mail, pager, or PDA if a database system is running out of memory or if a query is taking up too much time or processing power and allows them to make adjustments through any Web browser. There's also a new Configuration Advisor, which enables DBAs to reduce the time taken to configure databases and eliminate the need for frequent manual tuning of performance-related parameters. It sets parameters, based on a few questions, to help reduce the time to carry out these tasks to minutes compared with days.

An extension to DB2's federated capabilities includes new features that enable sites to integrate information available as Web services. Through a single SQL query, it can access and consolidate information from Web services providers without the need of going through applications.
Also, it's designed to enable programmers to speed up integration projects and benefit from a more automated way to link companies over the Internet.

Enhancements for XML are said to make it easier for programmers to integrate DB2 and XML information.

IBM has also built a Developer Center, enabling developers to build and deploy applications for either the Java or Windows platform.

For further information contact your local IBM representative.
URL: http://www.ibm.com/db2/v8beta.