127

DB2

May 2003

In this issue

update

DB2 Update

Published by

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

North American office

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

Subscriptions and back-issues

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

DB2 Update on-line

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

Editor

Trevor Eddolls

Disclaimer

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

Contributions

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

Printed in England.

Table functions

A user-defined table function is a function which returns a table to the SQL statement that references it. A table function can be referenced only in the FROM clause of a SELECT statement. In general, the returned table can be referenced in exactly the same way as any other table. Table functions are useful for performing SQL operations on non-DB2 data or moving non-DB2 data into a DB2 table.

EXAMPLE 1

Example 1 demonstrates the definition and use of a user-defined table function called ROWDUMMY. This function is written in PL/I and returns *n* rows. The function takes one input parameter (n) and returns a table of one column and n rows. The external name, ROWDUMMY, identifies the name of the load module containing the code for the function. The SELECT statement in the example shows how the ROWDUMMY function is invoked.

```
CREATE FUNCTION SYSADM.ROWDUMMY
       ( NUM  INTEGER )
RETURNS TABLE ( ROW INTEGER )
SPECIFIC ROWDUMMY
EXTERNAL NAME 'ROWDUMMY'
LANGUAGE PLI
PARAMETER STYLE DB2SQL
NOT DETERMINISTIC
READS SQL DATA
NO DBINFO
FENCED
NO COLLID
WLM ENVIRONMENT DSNNWLM
STAY RESIDENT NO
PROGRAM TYPE MAIN
EXTERNAL ACTION
CALLED ON NULL INPUT
SCRATCHPAD 100
FINAL CALL
DISALLOW PARALLEL
ASUTIME LIMIT 10
SECURITY DB2 ;
SELECT * FROM TABLE (SYSADM.ROWDUMMY(7) ) X
```

Result:

```
        ROW
-----------
          1
          2
          3
          4
          5
          6
          7
```

The ROWDUMMY table function returns 7 rows (increment by 1) because the input parameter is 7.

EXAMPLE 2

The query returns the third Thursday in every month in the year 2003. This query also demonstrates use of the ROWDUMMY table function.

```
SELECT MONTH (GEN_DATE) MONTH_OF_YEAR,
       GEN_DATE "THIRD THURSDAY"
FROM (
SELECT DATE ('2002-12-31') + ROW DAY GEN_DATE,
       DAYOFWEEK (DATE('2002-12-31') + ROW DAYS ) DAY_OF_WEEK,
       DAYOFMONTH(DATE('2002-12-31') + ROW DAYS ) DAY_OF_MONTH
FROM TABLE (SYSADM.ROWDUMMY(365)) X ) Y
WHERE DAY_OF_MONTH BETWEEN 15 AND 21
  AND DAY_OF_WEEK=5 ;
```

Result:

```
      MONTH
       OF
      YEAR   THIRD THURSDAY
----------  --------------
          1  2003-01-16
          2  2003-02-20
          3  2003-03-20
          4  2003-04-17
          5  2003-05-15
          6  2003-06-19
          7  2003-07-17
          8  2003-08-21
          9  2003-09-18
         10  2003-10-16
```

```
11  2003-11-20
12  2003-12-18
```

This query generates 365 rows, all days in year 2003. The *where* condition defines only requested data.

EXAMPLE 3

Example 3 demonstrates the definition and use of a user-defined table function called RSQL. This function is written in PL/I and returns a recursion report. The bill of material is now resolved in an SQL statement with a table function. The function takes seven input parameters and returns a table of five columns.

The input parameters are:

- *table creator* – VARCHAR(8)
- *table name* – VARCHAR(18)
- *parent_column* – VARCHAR(18)
- *parent_value* – VARCHAR(35)
- *dependent_column* – VARCHAR(18)
- *additional_column* – ARCHAR(18)
- *additional_column_value* – VARCHAR(35)
- *parameter_number* – INTEGER.

The output parameters are:

- *rid* – INTEGER * row id
- *level* – CHAR(9)
- *parent_key* – VARCHAR(35)
- *dependent_key* – VARCHAR(35)
- *message* – VARCHAR(70).

```
CREATE FUNCTION SYSADM.RSQL
       ( CREATOR            VARCHAR(8)
       , TBNAME             VARCHAR(18)
```

```
       , PARENT_COLUMN      VARCHAR(18)
       , PARENT_VALUE       VARCHAR(35)
       , DEPENDENT_COLUMN   VARCHAR(18)
       , ADDITIONAL_COLUMN  VARCHAR(18)
       , ADD_COL_VALUE      VARCHAR(35)
       , PARAMETER_NUMBER   INTEGER )
RETURNS TABLE
       ( RID                INTEGER
       , LEVEL              CHAR(9)
       , PARENT_KEY         VARCHAR(35)
       , DEPENDENT_KEY      VARCHAR(35)
       , MESSAGE            VARCHAR(70) )
SPECIFIC RSQL
EXTERNAL NAME 'RSQL'
LANGUAGE PLI
PARAMETER STYLE DB2SQL
NOT DETERMINISTIC
READS SQL DATA
NO DBINFO
FENCED
COLLID RSQL
WLM ENVIRONMENT DSNNWLM
STAY RESIDENT NO
PROGRAM TYPE SUB
EXTERNAL ACTION
CALLED ON NULL INPUT
SCRATCHPAD 100
FINAL CALL
DISALLOW PARALLEL
ASUTIME LIMIT 10
SECURITY DB2
;
```

The SELECT statement shows a recursive report on referential integrity on DB2 Catalog V7. The parent key on the SYSIBM.SYSRELS table is REFTBNAME and the dependent key is TBNAME. The init value for the parent key is SYSTABLESPACE. The additional column is CREATOR with value SYSIBM. For the last parameter, a value 1 switches on an additional column value, and a value 0 switches off an additional value.

```
SELECT RID,LEVEL,DEPENDENT_KEY,PARENT_KEY
FROM TABLE (
     SYSADM.RSQL('SYSIBM','SYSRELS','REFTBNAME',
                 'SYSTABLESPACE','TBNAME',
                 'CREATOR','SYSIBM',1
                 )
            ) X
```

Result:

```
                  DEPENDENT           PARENT
   RID  LEVEL        KEY                KEY
------  --------  -----------------  -----------------
     1  1         SYSTABLESPACE
     2   2        SYSTABLES          SYSTABLESPACE
     3    3       SYSCHECKS          SYSTABLES
     4     4      SYSCHECKDEP        SYSCHECKS
     5     4      SYSCHECKS2         SYSCHECKS
     6    3       SYSTABAUTH         SYSTABLES
     7     4      SYSCOLAUTH         SYSTABAUTH
     8    3       SYSSYNONYMS        SYSTABLES
     9    3       SYSRELS            SYSTABLES
    10     4      SYSLINKS           SYSRELS
    11     4      SYSFOREIGNKEYS     SYSRELS
    12    3       SYSRELS            SYSTABLES
    13     4      SYSLINKS           SYSRELS
    14     4      SYSFOREIGNKEYS     SYSRELS
    15    3       SYSINDEXES         SYSTABLES
    16     4      SYSKEYS            SYSINDEXES
    17     4      SYSINDEXPART       SYSINDEXES
    18      5     SYSINDEXPART_HIST  SYSINDEXPART
    19     4      SYSINDEXSTATS      SYSINDEXES
    20      5     SYSINDEXSTATS_HIST SYSINDEXSTATS
    21     4      SYSINDEXES_HIST    SYSINDEXES
    22    3       SYSCOLUMNS         SYSTABLES
    23     4      SYSFIELDS          SYSCOLUMNS
    24     4      SYSCOLSTATS        SYSCOLUMNS
    25     4      SYSCOLDISTSTATS    SYSCOLUMNS
    26     4      SYSCOLDIST         SYSCOLUMNS
    27     4      SYSCOLDIST_HIST    SYSCOLUMNS
    28     4      SYSCOLUMNS_HIST    SYSCOLUMNS
    29    3       SYSTABSTATS        SYSTABLES
    30     4      SYSTABSTATS_HIST   SYSTABSTATS
    31    3       SYSCONSTDEP        SYSTABLES
    32    3       SYSTRIGGERS        SYSTABLES
    33    3       SYSTABCONST        SYSTABLES
    34     4      SYSKEYCOLUSE       SYSTABCONST
    35    3       SYSTABLES_HIST     SYSTABLES
    36   2        SYSTABLEPART       SYSTABLESPACE
    37    3       SYSTABLEPART_HIST  SYSTABLEPART
    38   2        SYSLOBSTATS        SYSTABLESPACE
    39    3       SYSLOBSTATS_HIST   SYSLOBSTATS
```

ROWDUMMY PL/I SOURCE CODE

```
* PROCESS SYSTEM(MVS);
 ROWDMMY: PROC(NUM, ROW, NUM_INDR, ROW_INDR,
```

```
            UDF_SQLSTATE, UDF_NAME, UDF_SPEC_NAME,
            UDF_DIAG_MSG, UDF_SCRATCHPAD,
            UDF_CALL_TYPE, UDF_DBINFO)
        OPTIONS(FETCHABLE NOEXECOPS REENTRANT);
/*********************************************************************/
/*    UDF   : ROWDUMMY     ROW GENERATOR                             */
/*    INPUT : NUM          INTEGER                                    */
/*    OUTPUT: ROW          INTEGER   OUTPUT PARAMETER                */
/*********************************************************************/
 DCL NUM         BIN FIXED(31);        /* INPUT  PARAMETER          */
 DCL ROW         BIN FIXED(31);        /* RESULT PARAMETER          */
 DCL NUM_INDR    BIN FIXED(15);        /* INDIKATOR FOR INPUT       */
 DCL ROW_INDR    BIN FIXED(15);        /* INDIKATOR FOR RESULT      */
 DCL 1 UDF_SCRATCHPAD,                 /* SCRATCHPAD               */
      3 UDF_SPAD_LEN    BIN FIXED(31),
      3 UDF_SPAD_TEXT   CHAR(100),
       5 COUNTR         BIN FIXED(31);
 DCL 1 UDF_DBINFO,                          /* DBINFO               */
     3 UDF_DBINFO_LLEN BIN FIXED(15),       /* LOCATION LENGTH      */
     3 UDF_DBINFO_LOC  CHAR(128),           /* LOCATION NAME        */
     3 UDF_DBINFO_ALEN BIN FIXED(15),       /* AUTH ID LENGTH       */
     3 UDF_DBINFO_AUTH CHAR(128),           /* AUTHORIZATION ID     */
     3 UDF_DBINFO_CCSID CHAR(48),           /* CCSIDS FOR DB2 OS/390 */
      5 UDF_DBINFO_ESBCS BIN FIXED(31),     /* EBCDIC SBCS CCSID    */
      5 UDF_DBINFO_EMIXED BIN FIXED(31),    /* EBCDIC MIXED CCSID   */
      5 UDF_DBINFO_EDBCS BIN FIXED(31),     /* EBCDIC DBCS CCSID    */
      5 UDF_DBINFO_ASBCS BIN FIXED(31),     /* ASCII SBCS CCSID     */
      5 UDF_DBINFO_AMIXED BIN FIXED(31),    /* ASCII MIXED CCSID    */
      5 UDF_DBINFO_ADBCS BIN FIXED(31),     /* ASCII DBCS CCSID     */
      5 UDF_DBINFO_RESERV1 CHAR(20),        /* RESERVED             */
     3 UDF_DBINFO_QLEN BIN FIXED(15),       /* QUALIFIER LENGTH     */
     3 UDF_DBINFO_QUALIF CHAR(128),         /* QUALIFIER NAME       */
     3 UDF_DBINFO_TLEN BIN FIXED(15),       /* TABLE LENGTH         */
     3 UDF_DBINFO_TABLE  CHAR(128),         /* TABLE NAME           */
     3 UDF_DBINFO_CLEN BIN FIXED(15),       /* COLUMN LENGTH        */
     3 UDF_DBINFO_COLUMN CHAR(128),         /* COLUMN NAME          */
     3 UDF_DBINFO_RELVER CHAR(8),           /* DB2 RELEASE LEVEL    */
     3 UDF_DBINFO_PLATFORM BIN FIXED(31),   /* DATABASE PLATFORM    */
     3 UDF_DBINFO_NUMTFCOL BIN FIXED(15),   /* # OF TF COLS USED    */
     3 UDF_DBINFO_RESERV1 CHAR(24),         /* RESERVED             */
     3 UDF_DBINFO_TFCOLUMN PTR,             /* -> TABLE FUN COL LIST */
     3 UDF_DBINFO_RESERV2 CHAR(24);         /* RESERVED             */
 DCL (ADDR,LENGTH,SUBSTR,NULL)   BUILTIN;
 EXEC SQL INCLUDE SQLCA;
 DCL (LENGTH,ADDR,NULL)   BUILTIN;
 IF COUNTR>10000 THEN UDF_SQLSTATE='02000';
 IF UDF_CALL_TYPE=-2 THEN  COUNTR=0
 IF UDF_CALL_TYPE=-1 THEN ROW=0;
 IF UDF_CALL_TYPE=0 THEN DO;
    COUNTR=COUNTR+1;
```

```
    ROW=COUNTR;
    ROW_INDR=Ø;
    IF COUNTR>NUM THEN UDF_SQLSTATE='Ø2ØØØ';
  END;
  IF UDF_CALL_TYPE= 1 THEN  UDF_SQLSTATE='Ø2ØØØ';
  IF UDF_CALL_TYPE= 2 THEN  UDF_SQLSTATE='Ø2ØØØ';
  END;
 END ROWDMMY;
```

RSQL PL/I SOURCE CODE

```
* PROCESS SYSTEM(MVS);
 RSQLØØ: PROC(CREATOR, TBNAME, PARENT_COLUMN, PARENT_VALUE,
              DEPENDENT_COLUMN, ADDITIONAL_COLUMN, ADD_COL_VALUE,
              PARAMETER_NUMBER, RID, LEVEL, PARENT_KEY,
              DEPENDENT_KEY, MESSAGE,
              PR1_INDR, PR2_INDR, PR3_INDR, PR4_INDR,
              PR5_INDR, PR6_INDR, PR7_INDR, PR8_INDR,
              PR9_INDR, PR1Ø_INDR, PR11_INDR, PR12_INDR,
              PR13_INDR,
              UDF_SQLSTATE, UDF_NAME, UDF_SPEC_NAME,
              UDF_DIAG_MSG, UDF_SCRATCHPAD,
              UDF_CALL_TYPE, UDF_DBINFO)
           OPTIONS(FETCHABLE NOEXECOPS REENTRANT);
 /*****************************************************************/
 /*    UDF   : RSQL                      RECURSION SQL            */
 /*    INPUT : CREATOR                   VARCHAR(8)               */
 /*    INPUT : TBNAME                    VARCHAR(18)              */
 /*    INPUT : PARENT_COLUMN             VARCHAR(18)              */
 /*    INPUT : PARENT_VALUE              VARCHAR(35)              */
 /*    INPUT : DEPENDENT_COLUMN          VARCHAR(18)              */
 /*    INPUT : ADDITIONAL_COLUMN         VARCHAR(18)              */
 /*    INPUT : ADD_COL_VALUE             VARCHAR(35)              */
 /*    INPUT : PARAMETER_NUMBER          INTEGER                  */
 /*    OUTPUT: RID                       INTEGER                  */
 /*    OUTPUT: LEVEL                     CHAR(9)                  */
 /*    OUTPUT: PARENT_KEY                VARCHAR(35)              */
 /*    OUTPUT: DEPENDENT_KEY             VARCHAR(35)              */
 /*    OUTPUT: MESSAGE                   VARCHAR(7Ø)              */
 /*****************************************************************/
 /* INPUT DECLARATION                                             */
 /*****************************************************************/
  DCL CREATOR     CHAR(8) VAR;
  DCL (TBNAME, PARENT_COLUMN, DEPENDENT_COLUMN,
       ADDITIONAL_COLUMN)                    CHAR(18) VAR;
  DCL (PARENT_VALUE, ADD_COL_VALUE)          CHAR(35) VAR;
  DCL  PARAMETER_NUMBER                      BIN FIXED(31);
 /*****************************************************************/
 /* OUTPUT DECLARATION                                            */
```

```
/*****************************************************************/
 DCL RID                                      BIN FIXED(31);
 DCL LEVEL                                    CHAR(9);
 DCL (PARENT_KEY, DEPENDENT_KEY)              CHAR(35) VAR;
 DCL MESSAGE                                  CHAR(7Ø) VAR;
/*****************************************************************/
/* INDIKATOR VARIABLE                                            */
/*****************************************************************/
 DCL (PR1_INDR, PR2_INDR, PR3_INDR, PR4_INDR,
      PR5_INDR, PR6_INDR, PR7_INDR, PR8_INDR, PR9_INDR,
      PR1Ø_INDR, PR11_INDR, PR12_INDR, PR13_INDR) BIN FIXED(15);
/*****************************************************************/
 DCL 1 UDF_SCRATCHPAD,                /* SCRATCHPAD               */
      3 UDF_SPAD_LEN    BIN FIXED(31),
      3 UDF_SPAD_TEXT   CHAR(1ØØ),
       5 COUNTR          BIN FIXED(31);
 DCL 1 UDF_DBINFO,                        /* DBINFO               */
     3 UDF_DBINFO_LLEN BIN FIXED(15),     /* LOCATION LENGTH      */
     3 UDF_DBINFO_LOC  CHAR(128),         /* LOCATION NAME        */
     3 UDF_DBINFO_ALEN BIN FIXED(15),     /* AUTH ID LENGTH       */
     3 UDF_DBINFO_AUTH CHAR(128),         /* AUTHORIZATION ID     */
     3 UDF_DBINFO_CCSID CHAR(48),         /* CCSIDS FOR DB2 OS/39Ø */
      5 UDF_DBINFO_ESBCS BIN FIXED(31),   /* EBCDIC SBCS CCSID    */
      5 UDF_DBINFO_EMIXED BIN FIXED(31),  /* EBCDIC MIXED CCSID   */
      5 UDF_DBINFO_EDBCS BIN FIXED(31),   /* EBCDIC DBCS CCSID    */
      5 UDF_DBINFO_ASBCS BIN FIXED(31),   /* ASCII SBCS CCSID     */
      5 UDF_DBINFO_AMIXED BIN FIXED(31),  /* ASCII MIXED CCSID    */
      5 UDF_DBINFO_ADBCS BIN FIXED(31),   /* ASCII DBCS CCSID     */
      5 UDF_DBINFO_RESERV1 CHAR(2Ø),      /* RESERVED             */
     3 UDF_DBINFO_QLEN BIN FIXED(15),     /* QUALIFIER LENGTH     */
     3 UDF_DBINFO_QUALIF CHAR(128),       /* QUALIFIER NAME       */
     3 UDF_DBINFO_TLEN BIN FIXED(15),     /* TABLE LENGTH         */
     3 UDF_DBINFO_TABLE  CHAR(128),       /* TABLE NAME           */
     3 UDF_DBINFO_CLEN BIN FIXED(15),     /* COLUMN LENGTH        */
     3 UDF_DBINFO_COLUMN CHAR(128),       /* COLUMN NAME          */
     3 UDF_DBINFO_RELVER CHAR(8),         /* DB2 RELEASE LEVEL    */
     3 UDF_DBINFO_PLATFORM BIN FIXED(31), /* DATABASE PLATFORM    */
     3 UDF_DBINFO_NUMTFCOL BIN FIXED(15), /* # OF TF COLS USED    */
     3 UDF_DBINFO_RESERV1 CHAR(24),       /* RESERVED             */
     3 UDF_DBINFO_TFCOLUMN PTR,           /* -> TABLE FUN COL LIST */
     3 UDF_DBINFO_RESERV2 CHAR(24);       /* RESERVED             */
 EXEC SQL INCLUDE SQLCA;
/*****************************************************************/
/* OUTPUT HOST DECLARATION                                       */
/*****************************************************************/
 DCL (PKHV,DKHV) CHAR(35) VAR;
/*****************************************************************/
 DCL (LENGTH,SUBSTR,ADDR,NULL,SYSNULL)   BUILTIN;
 DCL MTEXT           CHAR(7Ø) STATIC;
 DCL ADDC            CHAR(35) VAR STATIC;
```

```
 IF PARAMETER_NUMBER=1
 THEN ADDC='  AND '!!ADDITIONAL_COLUMN!!'='''!!ADD_COL_VALUE!!''' ';
 ELSE ADDC=' ';
/*********************************************************************/
/* OUTPUT STATIC VARIABLES                                           */
/*********************************************************************/
 DCL RIDP(10000)       BIN FIXED(31) STATIC;
 DCL LEVP(10000)       CHAR(9)       STATIC;
 DCL PKEY(10000)       CHAR(35) VAR  STATIC;
 DCL DKEY(10000)       CHAR(35) VAR  STATIC;
 DCL MESS(10000)       CHAR(70) VAR  STATIC;
/*********************************************************************/
 DCL (K,IJ)            BIN FIXED(15) STATIC;
 DCL (QUERY1,QUERY2)   CHAR(255) VAR;
 IF COUNTR>10000 THEN UDF_SQLSTATE='02000';
/*********************************************************************/
/* FIRST CALL TO THE UDF                                             */
/*********************************************************************/
 IF UDF_CALL_TYPE=-2 THEN DO;
    DCL LEVELP      CHAR(9) INIT(' ');
    IJ=0;
    K=1;
    COUNTR=0;
    CALL RECURS(PARENT_VALUE,2);
    RECURS:PROC(PKV,LEV) RECURSIVE;
           DCL (I,NUMD)         BIN FIXED(31);
           DCL PKV              CHAR(35) VAR;
           DCL LEV              PIC '9';
           QUERY1='SELECT '!!PARENT_COLUMN!!','
                          !!DEPENDENT_COLUMN!!' '!!
                  'FROM ' !!CREATOR!!'.'!!TBNAME!!' '!!
                  'WHERE '!!PARENT_COLUMN!!'='''!!PKV!!''' '!!
                  '  AND '!!PARENT_COLUMN!!'<>'!!DEPENDENT_COLUMN!!
                   ADDC!!
                  'ORDER BY '!!PARENT_COLUMN!!' '!!
                  'WITH UR';
           EXEC SQL DECLARE C1 CURSOR FOR STMT1;
           EXEC SQL PREPARE STMT1 FROM QUERY1;
           QUERY2='SELECT COUNT(*) '!!
                  'FROM ' !!CREATOR!!'.'!!TBNAME!!' '!!
                  'WHERE '!!PARENT_COLUMN!!'='''!!PKV!!''' '!!
                  '  AND '!!PARENT_COLUMN!!'<>'!!DEPENDENT_COLUMN!!
                   ADDC!!
                  'WITH UR';
           EXEC SQL DECLARE C2 CURSOR FOR STMT2;
           EXEC SQL PREPARE STMT2 FROM QUERY2;
           EXEC SQL OPEN C2;
           EXEC SQL FETCH C2 INTO :NUMD;
           IF SQLCODE < 0 THEN DO;
              MTEXT='SQLCODE='!!SQLCODE;
```

```
            UDF_CALL_TYPE=Ø;
            K=Ø;
            GOTO L1;
         END;
         EXEC SQL CLOSE C2;
         IF NUMD = Ø THEN RETURN;         /* NO MORE ROWS   */
      BEGIN;
         DCL 1 RQ(NUMD),
             2 PK  CHAR(35) VAR,
             2 DK  CHAR(35) VAR;
         EXEC SQL OPEN C1;
         DO I=1 TO NUMD;
            IF I > 10000 THEN LEAVE;
            EXEC SQL FETCH C1 INTO :PKHV, :DKHV;
            PK(I) = PKHV;
            DK(I) = DKHV;
         END;
         EXEC SQL CLOSE C1;
         DO I=1 TO NUMD;
            IF I > 10000 THEN LEAVE;
            K=K+1;
            RIDP(K)=K;
            SUBSTR(LEVELP,LEV,1)=LEV;
            LEVP(K)=LEVELP;
            PKEY(K)=PK(I);
            DKEY(K)=DK(I);
            LEVELP=' ';
            CALL RECURS(DKEY(K),LEV+1);
         END;
      END;
    END RECURS;
 END;
/*********************************************************************/
/* FETCH CALL TO THE UDF                                             */
/*********************************************************************/
 IF UDF_CALL_TYPE=Ø THEN DO;
    COUNTR=COUNTR+1;
    IJ=IJ+1;
    IF IJ<=K THEN DO;
       IF IJ=1 THEN DO;
          RID=IJ;
          LEVEL='1';
          PARENT_KEY=' ';
          DEPENDENT_KEY=PARENT_VALUE;
          MESSAGE='PARENT KEY N/A';
       END;
       ELSE DO;
          RID=RIDP(IJ);
          LEVEL=LEVP(IJ);
          PARENT_KEY=PKEY(IJ);
```

```
            DEPENDENT_KEY=DKEY(IJ);
            MESSAGE=' ';
         END;
      END;
      IF K=Ø THEN DO;
         RID=1;
         LEVEL=' ';
         PARENT_KEY=' ';
         DEPENDENT_KEY=' ';
         MESSAGE=MTEXT;
         K,IJ=1;
      END;
      IF IJ>9999 ! IJ>K THEN UDF_SQLSTATE='Ø2ØØØ';
   END;
  /*********************************************************************/
  /* FINAL CALL                                                        */
  /*********************************************************************/
  L1:
   IF UDF_CALL_TYPE= 2 THEN UDF_SQLSTATE='Ø2ØØØ';

  END RSQLØØ;
```

When using table functions, there are a few additional things that you have to know. Table functions do not use the same mechanism to pass information back to the program. The program that is invoking the table function (QMF for Windows in the example above) does its normal SQL processing as with any other regular table. It will open the cursor, fetch rows from it, and close the cursor. When executing open, fetch, and close calls, a trip is made to the UDF program, which executes in a WLM address space. On each trip to the UDF program, a CALL_TYPE parameter is passed to the (PL/I) program that is invoked. The program uses this information to decide what part of the code to execute. The possible CALL_TYPEs are:

- -2 – this is the first call to the user-defined table function.
- -1 – this is an open call to the user-defined table function by an sql statement.
- 0 – this is a fetch call. You will have multiple calls of this type, one for every row you want to pass back in the result table.
- 1 – this is a close call.
- 2 – this is a final call.

When you want to signal to the calling application that you have finished passing rows (end of rows), you set the UDF_SQLSTATE variable to '02000' before returning to the invoker.

Bernard Zver
DBA
Informatika (Slovenia)

Monitor tablespace and indexspace extents

I have developed a utility to monitor tablespace and indexspace extents periodically. It works for OS/390 2.10, IBM PL/I for MVS and VM Version 1.1.1, and DB2 5.1.2.

Although it is developed with DB2 V5, I believe it works for V6 and V7 too because the heart of this feature is a couple of SQL queries.

I used performance queries 20 and 21 in the DSN510.SDSNSAMP (DSNTESP) member to calculate the extents of the pagesets. The very important prerequisite for this utility is that your statistics must be current. It uses DB2 catalog statistics for the tables below:

- SYSIBM.SYSTABLESPACE (DBNAME, NAME)
- SYSIBM.SYSTABLEPART (PARTITION, SPACE, PQTY, SQTY)
- SYSIBM.SYSINDEXES (CREATOR, NAME)
- SYSIBM.SYSINDEXPART (PARTITION, SPACE, PQTY, SQTY).

I have modified the original queries and added some more functions:

1 The original queries supplied from IBM do not calculate the new primary quantity values. I calculate the new PQTY value based on #extents, and old primary and secondary values.

2 I create another dataset and append ALTER TABLESPACE and ALTER INDEX statements with the new primary quantity values, ready to run through SPUFI, DSNTEP2, or any other SQL processor.

3 A threshold value for reporting the extents can easily be given in the SYSIN DD statement of the DB2EXTJ.

The main program is a PL/I-DB2 program, which runs those queries against the DB2 catalog and calculates, formats, and reports the sample output to a dataset.

I have included part of the sample report:

```
---------------------------------------------
DB2 Extent Automation Report
Copyright (c)
Created : 2002-10-23-15.50.59.182275
Alter Statements: BAGKUR.DB2.EXTENTS.ALTERS
---------------------------------------------

Database  Tablespace  Part Ext  New PQTY
--------  ----------  ---- --  --------
DBDENE    TSDENE      00   28   000000000348
DBKARNE   TSKARLOG    00   24   000003734340
DBECRDN   TSECREDE    00   16   000001040000
DBGOZL    TSGOZIPT    00   10   000000010208
DBKARNE   TSKARNE     02   10   000000400000
DBKARNE   TSKARNE     01   10   000000400000

Database Index              Part Ext New PQTY
-------- ----------------- ---- -- --------
OLTAR    BAGGEC             00   35  000000000880
SYSIBM   DSNKSX01           00   31  000000046080
CIHAZ    BAGYAK             00   29  000000019784
OLUM     BAGYAK             00   29  000000019784
SAGLIK   BAGYAK             00   29  000000019784
```

Some views from the ALTER statement's dataset are shown below:

```
ALTER TABLESPACE DBDENE  .TSDENE            PRIQTY 000000000348;
ALTER TABLESPACE DBKARNE .TSKARLOG          PRIQTY 000003734340;
ALTER TABLESPACE DBECRDN .TSECREDE          PRIQTY 000001040000;
ALTER TABLESPACE DBGOZL  .TSGOZIPT          PRIQTY 000000010208;
ALTER TABLESPACE DBKARNE .TSKARNE PART 2    PRIQTY 000000400000;
ALTER TABLESPACE DBKARNE .TSKARNE PART 1    PRIQTY 000000400000;
ALTER INDEX OLTAR        .BAGGEC            PRIQTY 000000000880;
```

```
ALTER INDEX SYSIBM       .DSNKSX01          PRIQTY 000000046080;
ALTER INDEX CIHAZ        .BAGYAK            PRIQTY 000000019784;
ALTER INDEX OLUM         .BAGYAK            PRIQTY 000000019784;
ALTER INDEX SAGLIK       .BAGYAK            PRIQTY 000000019784;
```

If you do not have any ISV software or anything else to monitor your extents, I suggest that you use this utility. You can easily enhance it by adding REORG features after ALTERs, or let your DBAs do it.

The DB2EXTJ JCL is for running the DB2EXT program. Look at the SYSIN DD in step BAGKUR. It uses this value to report extents that are greater than this value.

```
//DB2EXTJ  JOB (CUNEYT,EXP),'CUNEYT',NOTIFY=&SYSUID,REGION=2M,
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//*
//*  DB2 Extent Automation.
//*  Mehmet Cuneyt GOKSU
//*
//DELETE   EXEC PGM=IDCAMS,COND=(8,LT)
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  DELETE BAGKUR.DB2.EXTENTS
  DELETE BAGKUR.DB2.EXTENTS.ALTERS
  SET MAXCC = 0
//*
//BAGKUR   EXEC PGM=IKJEFT01,COND=(8,LT)
//EXTOUT   DD DSN=BAGKUR.DB2.EXTENTS,
//            DISP=(NEW,CATLG),UNIT=3390,
//            SPACE=(TRK,(1,1),RLSE),
//            DCB=(RECFM=VB,LRECL=132,BLKSIZE=1320)
//ALTOUT   DD DSN=BAGKUR.DB2.EXTENTS.ALTERS,
//            DISP=(NEW,CATLG),UNIT=3390,
//            SPACE=(TRK,(1,1),RLSE),
//            DCB=(RECFM=VB,LRECL=132,BLKSIZE=1320)
//SYSOUT    DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//SYSDBOUT  DD SYSOUT=*
//SYSABOUT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD *
  DSN SYSTEM(DB2T)
  RUN PROGRAM(DB2EXT) PLAN(DB2EXT) -
      LIBRARY('BAGKUR.BATCH.LOADLIB')
END
//SYSIN     DD *
5             <=== Extent Threashold
```

```
/*
//*
```

PL/I SOURCE FOR DB2EXT

```
* PROCESS XREF,ATTRIBUTES,MARGINI('!'),WORKFILE(3340),GOSTMT,AGGREGATE,
       MAP,INCLUDE,STG,SIZE(MAX),NEST;
  DB2EXT: PROC OPTIONS(MAIN);
 /*------------------------------------------------------------*/
   DCL 1 REPORT1,
       2 WS_FILLER1            CHAR (02) INIT('  '),
       2 WS_DBNAME             CHAR (08),
       2 WS_FILLER2            CHAR (02) INIT('  '),
       2 WS_TSNAME             CHAR (08),
       2 WS_FILLER3            CHAR (04) INIT('    '),
       2 WSS_PARTITION         PIC'(2)9',
       2 WS_FILLER4            CHAR (03) INIT('   '),
       2 WSS_EXTENT            PIC'(2)9',
       2 WS_FILLER5            CHAR (03) INIT('   '),
       2 WSS_NEWPQTY           PIC'(12)9';
   DCL 1 REPORT2,
       2 WS_FILLER1            CHAR (02) INIT('  '),
       2 WS_CREATOR            CHAR (08) INIT(' '),
       2 WS_FILLER2            CHAR (01) INIT(' '),
       2 WS_IXNAME             CHAR (18) INIT (' '),
       2 WS_FILLER3            CHAR (01) INIT(' '),
       2 WSS_PARTITION_IX      PIC'(2)9',
       2 WS_FILLER4            CHAR (03) INIT(' '),
       2 WSS_EXTENT_IX         PIC'(2)9',
       2 WS_FILLER5            CHAR (02) INIT(' '),
       2 WSS_NEWPQTY_IX        PIC'(12)9';
   DCL 1 YAPI,
       2 WS_SPACE              BIN FIXED(31),
       2 WS_PQTY               BIN FIXED(31),
       2 WS_SQTY               BIN FIXED(15),
       2 WS_PARTITION          BIN FIXED(15),
       2 WS_EXTENT             BIN FIXED(15),
       2 WS_NEWPQTY            BIN FIXED(31);
   DCL
      1  TEXT_MSG              CHAR (80),
      1  SAY                   BIN FIXED(15),
      1  I                     BIN FIXED(15),
      1  EXTENT_INPUT          BIN FIXED(15),
      1  EXTENT_INPUT_S        CHAR (2) ,
      1  WRKTIME               CHAR(26) ;
   DCL  EXTOUT   FILE RECORD SEQL OUTPUT;
   DCL  ALTOUT   FILE RECORD SEQL OUTPUT;
  /*----------C U R S O R S -----------------------------------**/
           EXEC SQL INCLUDE SQLCA ;
```

```
 EXEC SQL DECLARE C1 CURSOR FOR
SELECT * FROM
  (SELECT
     TS.DBNAME AS DB,
     TS.NAME AS TS,
     TP.PARTITION AS PART,
     TP.SPACE,
     TP.PQTY,
     TP.SQTY,
     CASE TS.PGSIZE
       WHEN 4 THEN
          (TP.SPACE-
            CASE
              WHEN TP.PQTY<3 THEN 12
              ELSE TP.PQTY*4
            END)
            / (CASE
                 WHEN TP.SQTY=Ø THEN 1
                 WHEN TP.SQTY<3 THEN 12
                 ELSE TP.SQTY*4
               END) +1
       WHEN 8 THEN
          (TP.SPACE-
            CASE
              WHEN TP.PQTY<6 THEN 24
              ELSE TP.PQTY*4
            END)
            / (CASE
                 WHEN TP.SQTY=Ø THEN 1
                 WHEN TP.SQTY<6 THEN 24
                 ELSE TP.SQTY*4
               END) +1
       WHEN 16 THEN
          (TP.SPACE-
            CASE
              WHEN TP.PQTY<12 THEN 48
              ELSE TP.PQTY*4
            END)
            / (CASE
                 WHEN TP.SQTY=Ø THEN 1
                 WHEN TP.SQTY<12 THEN 48
                 ELSE TP.SQTY*4
               END) +1
       WHEN 32 THEN
          (TP.SPACE-
            CASE
              WHEN TP.PQTY<24 THEN 96
              ELSE TP.PQTY*4
            END)
            / (CASE
```

```
                    WHEN TP.SQTY=Ø THEN 1
                    WHEN TP.SQTY<24 THEN 96
                    ELSE TP.SQTY*4
                  END) +1
          END AS $#EXTENTS$
        FROM SYSIBM.SYSTABLESPACE TS,
             SYSIBM.SYSTABLEPART TP
        WHERE TS.NAME = TP.TSNAME AND
              TS.DBNAME = TP.DBNAME AND
              TP.SPACE>Ø)
        AS DUMMY
      WHERE #EXTENTS>:EXTENT_INPUT
      ORDER BY #EXTENTS DESC ;

      EXEC SQL DECLARE C2 CURSOR FOR
      SELECT * FROM
        (SELECT DISTINCT
           IX.CREATOR AS CR,
           IX.NAME AS IX,
           IP.PARTITION AS PART,
           IP.SPACE,
           IP.PQTY,IP.SQTY,
           CASE IX.PGSIZE
             WHEN 4Ø96 THEN
                (IP.SPACE-
                  CASE
                    WHEN IP.PQTY<3 THEN 48
                    ELSE IP.PQTY*4
                  END)
                  / (CASE
                       WHEN IP.SQTY=Ø THEN 1
                       WHEN IP.SQTY<3 THEN 48
                       ELSE IP.SQTY*4
                     END) +1
           END AS $#EXTENTS$
         FROM SYSIBM.SYSTABLESPACE TS, SYSIBM.SYSTABLES  TB,
              SYSIBM.SYSINDEXES IX, SYSIBM.SYSINDEXPART IP
         WHERE IX.NAME = IP.IXNAME AND
               TB.NAME = IX.TBNAME AND
               TS.NAME = TB.TSNAME AND
               IP.SPACE > Ø
         ) AS DUMMY
       WHERE #EXTENTS>:EXTENT_INPUT
       ORDER BY #EXTENTS DESC ;
EXEC SQL
SELECT CURRENT TIMESTAMP INTO :WRKTIME FROM SYSIBM.SYSDUMMY1;
PUT EDIT('-----------------------------------------')(COL(1),A);
PUT EDIT('Monitor extents                          ')(COL(1),A);
PUT EDIT('Copyright (c)                            ')(COL(1),A);
PUT EDIT('-----------------------------------------')(COL(1),A);
```

```
 PUT EDIT('                                                ')(COL(1),A);
SQLCODE=Ø;
SAY=Ø;
I=1;
 TEXT_MSG = '  ------------------------------------------------';
 WRITE FILE(EXTOUT) FROM(TEXT_MSG) ;
 TEXT_MSG = '  DB2 Extent Automation Report                     ';
 WRITE FILE(EXTOUT) FROM(TEXT_MSG) ;
 TEXT_MSG = '  Copyright (c)                                    ';
 WRITE FILE(EXTOUT) FROM(TEXT_MSG) ;
 TEXT_MSG = '  Created : '!!WRKTIME ;
 WRITE FILE(EXTOUT) FROM(TEXT_MSG) ;
 TEXT_MSG = '  Alter Statements: BAGKUR.DB2.EXTENTS.ALTERS     ';
 WRITE FILE(EXTOUT) FROM(TEXT_MSG) ;
 TEXT_MSG = '  ------------------------------------------------';
 WRITE FILE(EXTOUT) FROM(TEXT_MSG) ;
 TEXT_MSG = '                                                  ';
 WRITE FILE(EXTOUT) FROM(TEXT_MSG) ;
 TEXT_MSG = '  Database  Tablespace  Part Ext  New  PQTY       ';
 WRITE FILE(EXTOUT) FROM(TEXT_MSG) ;
 TEXT_MSG = '  --------  ----------  ---- --  --------       ';
 WRITE FILE(EXTOUT) FROM(TEXT_MSG) ;
 GET FILE(SYSIN) EDIT(EXTENT_INPUT_S) (COLUMN(1),A(2));
 EXTENT_INPUT = EXTENT_INPUT_S ;
   EXEC SQL OPEN C1;
      DO WHILE(SQLCODE=Ø);
         EXEC SQL FETCH C1 INTO
          :WS_DBNAME
         ,:WS_TSNAME
         ,:WS_PARTITION
         ,:WS_SPACE
         ,:WS_PQTY
         ,:WS_SQTY
         ,:WS_EXTENT ;
     IF SQLCODE=1ØØ THEN GO TO EXT_TS;
     SAY = SAY + 1;
     PUT EDIT('==> ',WS_DBNAME,' ',WS_TSNAME,' ',
         WS_PARTITION,' ',WS_SPACE,' ',WS_PQTY,' ',
         WS_SQTY,' ',WS_EXTENT)
         (COL(1),A,A,A,A,A,A,A,A,A,A,A,A,A,A);
    WS_NEWPQTY    = (WS_PQTY * 4) + (WS_SQTY * 4 * WS_EXTENT) ;
    WSS_PARTITION = WS_PARTITION ;
    WSS_EXTENT    = WS_EXTENT ;
    WSS_NEWPQTY   = WS_NEWPQTY ;
    WRITE FILE(EXTOUT) FROM(REPORT1);
    IF WS_PARTITION = Ø THEN DO ;
 TEXT_MSG = '  ALTER TABLESPACE '!!WS_DBNAME!!'.'!!WS_TSNAME!!
 ' PRIQTY '!!WSS_NEWPQTY!!';';
    END;
    ELSE DO;
```

```
TEXT_MSG = '  ALTER TABLESPACE '!!WS_DBNAME!!'.'!!WS_TSNAME!!
'PART '!!WS_PARTITION!!' PRIQTY '!!WSS_NEWPQTY!!';';
   END;
WRITE FILE(ALTOUT) FROM(TEXT_MSG) ;
     END;
EXT_TS:
   PUT EDIT('               ') (COL(1),A);
   PUT EDIT('Tablespace :   ',SAY) (COL(1),A,P'ZZZZZ');
   PUT EDIT('               ') (COL(1),A);
   EXEC SQL CLOSE C1;
TEXT_MSG = '  ----------------------------------------------';
WRITE FILE(EXTOUT) FROM(TEXT_MSG) ;
TEXT_MSG = '                                                ';
WRITE FILE(EXTOUT) FROM(TEXT_MSG) ;
TEXT_MSG = '  Database Index             Part Ext New  PQTY ';
WRITE FILE(EXTOUT) FROM(TEXT_MSG) ;
TEXT_MSG = '  -------- ---------------- ---- -- -------- ';
WRITE FILE(EXTOUT) FROM(TEXT_MSG) ;
SAY=Ø;
  EXEC SQL OPEN C2;
     DO WHILE(SQLCODE=Ø);
        EXEC SQL FETCH C2 INTO
         :WS_CREATOR
        ,:WS_IXNAME
        ,:WS_PARTITION
        ,:WS_SPACE
        ,:WS_PQTY
        ,:WS_SQTY
        ,:WS_EXTENT ;
    IF SQLCODE=1ØØ THEN GO TO EXT_IX;
    SAY = SAY + 1;
    PUT EDIT('==> ',WS_CREATOR,' ',WS_IXNAME,' ',
        WS_PARTITION,' ',WS_SPACE,' ',WS_PQTY,' ',
        WS_SQTY,' ',WS_EXTENT)
        (COL(1),A,A,A,A,A,A,A,A,A,A,A,A,A);
   WS_NEWPQTY    = (WS_PQTY * 4) + (WS_SQTY * 4 * WS_EXTENT) ;
   WSS_PARTITION_IX = WS_PARTITION ;
   WSS_EXTENT_IX    = WS_EXTENT ;
   WSS_NEWPQTY_IX   = WS_NEWPQTY ;
   WRITE FILE(EXTOUT) FROM(REPORT2);
   IF WS_PARTITION = Ø THEN DO;
TEXT_MSG = '  ALTER INDEX '!!WS_CREATOR!!'.'!!WS_IXNAME!!
' PRIQTY '!!WSS_NEWPQTY_IX!!';';
   END;
   ELSE DO;
TEXT_MSG = '  ALTER INDEX '!!WS_CREATOR!!'.'!!WS_IXNAME!!
' PART '!!WS_PARTITION!!' PRIQTY '!!WSS_NEWPQTY_IX!!';';
   END;
WRITE FILE(ALTOUT) FROM(TEXT_MSG) ;
     END;
```

```
   EXT_IX:
      PUT EDIT('               ') (COL(1),A);
      PUT EDIT('Index:         ',SAY) (COL(1),A,P'ZZZZZ');
      PUT EDIT('               ') (COL(1),A);
      EXEC SQL CLOSE C2;
  SON :
  END DB2EXT;
```

Mehmet Cuneyt Goksu
DB2 Specialist (Turkey)

DataPropagator user experiences and expectations on MVS – part 2

This month we conclude our look at techniques, including some undocumented features, to improve the performance of DataPropagator.

AVOIDING DUPLICATE CAPTURE OR APPLY TASKS

In Dprop Version 5, 6, or 7, both Apply and Capture tasks use exclusive MVS enqueues to prevent multiple instances running simultaneously.

An example for Apply is:

```
Major name = ASNAPPLY
Minor name = ASNAPP1D2O1
```

where *minor name* is ASN, apply-qualifier, DB2 subsystem.

Thus only one Apply is permitted to run for a single Apply qualifier.

An example for Capture is:

```
Major name = ASNLRPGM
Minor name = DSND2HØ    'C4E2D5C4F2C8FØ4Ø'x
```

where *minor name* has the DB2 group-id of a data-sharing group, or the subsystem ID of a non-data-sharing DB2.

Thus only one Capture can run.

If you start a duplicate Capture or Apply task, it will see the existing MVS enqueue and immediately end with an explanatory error message.

In Dprop V8, the MVS enqueues are no longer used. Two new control tables, IBMSNAP_APPENQ and IBMSNAP_CAPENQ, are used instead. The Apply or Capture task has a locking thread running as a separate MVS subtask, which presumably writes a row into their table but doesn't COMMIT it; and when the main task ends a ROLLBACK is done. That method enables exclusive locks, which can also be recognized from non-MVS systems.

The other difference for Dprop Version 8 is that you may have multiple sets of control tables for Capture (with different schemas), thus allowing multiple Captures to run on a single DB2 subsystem or on multiple members of a data-sharing group. This can be used to get throughput improvements of 50% or more with two or three Capture tasks (but with extra CPU usage).

DATA COMPRESSION

You can compress the CD and UOW tables. That could reduce DB2 logging on the Capture subsystem and potentially I/Os to and from bufferpools.

However, it only helps if you can create useful compression dictionaries, and that depends on the nature of the updates that you are replicating.

Note that the changes are still sent uncompressed across the network.

DB2 LOGS

The DB2 log output buffer should be made large on the Capture subsystem via DSNZPARM: OUTBUFF. Up to DB2 Version 5 the maximum was 4000, and from DB2 Version 6 the maximum is 400,000. If Capture is running on a member of a data-sharing

group, it reads the buffer directly on the local subsystem and reads the log files of all the others.

The log files and archive log files should also be made as large as reasonable, to allow for times when Capture has to read old log data.

If it has to read from the archive logs it can read them faster when they are on disk rather than tape. Therefore, we set up each of our log and archive log files to fill a disk volume, and use DFHSM to migrate and recall the archive logs as required. If you have virtual tapes, they could be a good alternative for the archive logs.

DB2 SORT POOL

Dprop V7 can do some large sorts, so you should have a large Sort pool. The maximum size is 64MB. Dprop V8 will hopefully reduce that need.

MONITORING REPLICATION

IBM offers the following command to show the position of Capture reading the DB2 log(s):

```
F capture-task,GETLSEQ
```

and you can tell by the timestamp in the resulting ASN0125I message how far it lags behind.

Alternatively, here is a query to show the lag of Capture (in log reading):

```
SELECT  DAY    (CURRENT TIMESTAMP - SYNCHTIME) * 86400
     +  HOUR   (CURRENT_TIMESTAMP - SYNCHTIME) * 3600
     +  MINUTE (CURRENT_TIMESTAMP - SYNCHTIME) * 60
     +  SECOND (CURRENT_TIMESTAMP - SYNCHTIME)
AS CAPTURE_LAG_SECS
FROM ASN.IBMSNAP_REGISTER
   WHERE GLOBAL_RECORD = 'Y'
WITH UR ;
```

Checking the Apply lag is a bit more complex, so I wrote a simple monitor to run as an MVS started task. That monitor is presented in an article in *Monitoring DataPropagator on MVS* in *DB2*

Update, issues 120 and 121 (October and November 2002). It is designed to monitor the progress of a single Dprop subscription set and warn if it gets too far behind. It builds a report showing snapshots of the propagation lag in Capture and Apply at regular intervals, plus the volume of changes and number of Apply cycles. If propagation gets too far behind it can issue warning messages and alert appropriate people. These warning messages can be used to trigger an automation tool to create further alerts too.

There is also DB2 Replication Monitoring Center, which is written by Thomas Eppner from IBM Global Services in Germany. It runs on NT or AIX and has much more functionality showing much detail about the active Dprop subscriptions generally, but it requires connectivity to the MVS systems. It also has an Alert system to detect when propagation gets behind, but it does not do that as well as my simple monitor. Just send an e-mail to eppner@de.ibm.com to get a copy of it.

Version 8 of DB2 UDB introduces Replication Alert Monitor, which can be started and checked from Replication Center (which is the new dropdown from Control Center), but until then no official IBM monitor is available. Dprop V8 includes many new control tables that will hold data for the new monitor, and it will be quite simple to write your own queries to check the status of Dprop, getting much more information than is currently possible. Otherwise, to check what Dprop is doing it is often useful to use a normal DB2 monitor to check what DB2 SQL statements are active for Apply and/or Capture tasks.

AUTOMATION FOR DPROP

MVS System Automation must synchronize the starting of the two started tasks, Capture and Apply. It should monitor whether they are running. It should also monitor messages and send alerts whenever problems occur.

We also run the Dprop monitor that I wrote. Automation starts and stops the monitor and it reacts to any error messages in the

MVS syslog from the monitor, alerting the people responsible whenever replication gets too far behind.

APPLY TRAIL TABLE

The apply trail table holds one row of statistics for each subscription set cycle. It can be improved by adding a column to show the time each cycle ended via the following:

- Adding an ENDTIME column to APPLYTRAIL (to hold CURRENT TIMESTAMP)
- Then subscription set process time = ENDTIME - LASTRUN

```
ALTER TABLE ASN.IBMSNAP_APPLYTRAIL
   ADD ENDTIME TIMESTAMP NOT NULL
               WITH DEFAULT;
```

IBM has also added an ENDTIME column, just like the above, to this table in Dprop V8 (along with some other new columns). When you have the ENDTIME column you can tell exactly how long each cycle took.

It is also worth adding an index, for example:

```
CREATE TYPE 2 INDEX ASN.IBMSNAP_APPLYTRAIX
       ON ASN.IBMSNAP_APPLYTRAIL
     ( LASTRUN              ASC  )
       USING STOGROUP grpname
             PRIQTY    pppp
             SECQTY    ssss
       BUFFERPOOL BPx ;
```

Then you should include LASTRUN in your SELECT predicate for queries of this table.

Apply trail needs pruning regularly. If you are replicating as fast as possible, using very many Apply cycles, there can be many cycles with nothing to do – creating rows with all ‘zero’ values plus the various timestamps. Here are two methods of pruning it, keeping error rows or those less than four days old.

With SQL:

- Delete old or ‘zero’ rows from the Apply trail:

```
SET CURRENT SQLID = 'xxxxxxxx';
ALTER TABLESPACE dbname.tsname LOCKSIZE TABLE;
COMMIT;
DELETE FROM ASN.IBMSNAP_APPLYTRAIL
  WHERE
    LASTRUN < CURRENT TIMESTAMP - 3 DAYS OR
    EFFECTIVE_MEMBERS =  Ø ;
ALTER TABLESPACE dbname.tsname LOCKSIZE ROW;
```

- Note that you need to change the locksize. This method includes much DB2 logging, so it is not recommended.

With DB2, the REORG utility discards unwanted rows, without any DB2 logging:

```
REORG TABLESPACE dbname.tsname LOG NO UNLOAD CONTINUE
SHRLEVEL NONE COPYDDN(SYSCOPY1)
STATISTICS TABLE (ALL) INDEX (ALL)
DISCARD FROM TABLE  ASN.IBMSNAP_APPLYTRAIL
WHEN ( (LASTRUN  < CURRENT TIMESTAMP - 3 DAYS) OR
       (EFFECTIVE_MEMBERS =  Ø ) )
```

Don’t worry too much about making this table unavailable while Apply is running. If that happens, Apply merely writes an error message at the end of a cycle when it tries to add a row to the table, and all the applied target table changes are committed and not rolled back.

CONCLUSION

How did our project perform? Initially it was far too slow, getting exponentially behind until single Apply cycles were running for over one hour! After we applied the techniques described here our Dprop was propagating at the high rate of more than 20,000 changes per minute, with the target table updates lagging only 10 to 15 seconds behind the source system. And if Capture and Apply are stopped for a while, they can catch up again at the rate of one hour of changes in about 15 minutes, indicating a maximum rate of over 60,000 changes per minute.

As you can see there are many tricks for Dprop V7, and the tuning tips can be extremely important when you replicate a large volume of changes.

In addition, many of the changes in Dprop V8 should make it much better, so we can look forward to improved performance and usability.

Ron Brown (Consultant)

Summary tables help improve the speed of large transactional queries

EXECUTIVE SUMMARY

Ask a production supervisor, accounts personnel dealing with totals in a financial system, a call centre supervisor, or anyone running hourly production reports or up-to-date queries, and you are almost sure to hear about the poor performance or slowness of the report because of the sheer bulk of raw data, which is increasing every day.

There is a solution to this – summary tables. This article describes how to use summary tables to hold collated information and have them updated as required. Summary tables can be used explicitly or automatically to improve the speed of certain queries.

Summary tables can be:

- Maintained by application logic or triggers.
- Automatic – maintained synchronously by DB2.
- Semi-automatic – maintained asynchronously by DB2 (updates upon request).
- Externally hosted – take the data elsewhere for analysis and reporting. Where you have a lot of activity on your source data, it may be worth replicating to another machine. In fact, this is recommended practice for on-line analytical processing (OLAP), but outside the scope of this article.

The first step is to decide how to group the data, so let us get some specifics on a domestic product production line. Adam supervises production of the product. The product is made in various colours (say, four), and sold by weight, in bags. Due to variations, each bag of product must be weighed, and stamped with a serial number. A bag of product is filled each minute, from each of four machines. The batch number and date must also be recorded for each bag. The listing below creates a table to record the domestic product production.

```
-- Don't forget to connect to a database first, connect to dbname user x
using pwd
CONNECT TO SAMPLE @
CREATE TABLE TEST.DOM_PRODUCT (
  Machine SMALLINT NOT NULL,
  SerialNo CHAR(1Ø) NOT NULL,
  Color CHAR(1Ø) NOT NULL,
  Weight SMALLINT NOT NULL,
  DateMade DATE NOT NULL,
  PRIMARY KEY (SerialNo)) @

-- Note: I am using "@" to terminate statements.
-- You can set this in the Command Center's
-- "Tools - Settings" dialog, in the "general" tab
Otherwise, enter commands individually
```

Now let's take three queries Adam must answer:

A What's the average bag weight for each colour from each machine overall?

B How many products of each colour have we produced in the past seven days excluding today, in how many bags, and with what average bag weight?

C How many products of each colour have we produced so far today, and in how many bags (must be up-to-the-minute)?

As the database stands, each of these queries could be answered with some simple SQL (see below), though the database would do a table scan for each query. This is much like Adam looking through every production report he's ever received.

Ordinary queries:

```
-- Query A
SELECT Machine,
  SUM(Weight)/COUNT(*) as AveWeight
FROM TEST.DOM_PRODUCT
GROUP by Machine @

-- Query B
SELECT Color,
  SUM(Weight) as SumWeight,
  COUNT(*) as Bags,
  SUM(Weight)/COUNT(*) as AveWeight
FROM TEST.DOM_PRODUCT
WHERE DateMade BETWEEN (current date - 7 days)
  AND (current date - 1 day)
GROUP BY Color @

-- Query C
SELECT Color,
  COUNT(*) as Bags,
  SUM(Weight) as SumWeight
FROM TEST.DOM_PRODUCT
WHERE DateMade = current date
GROUP BY Color @

-- Indexing can substantially improve performance
create index TEST.PRDMCIX ON  TEST.DOM_PRODUCT
(Machine, Color) @
create index TEST.PRDDMCIX ON  TEST.DOM_PRODUCT
(DateMade, Machine, Color) @

-- Use RUNSTATS when the table is loaded, so the database can decide
when to use an index

RUNSTATS ON TABLE TEST.DOM_PRODUCT WITH DISTRIBUTION AND
DETAILED INDEXES ALL @
```

Adding an index or two (also in the listing above) will help with queries B and C, like Adam filing his papers by date, but it's still a lot of reading.

Our raw data comprises more than 5,000 records per day of production (4 machines * 60 minutes * 24 hours = 5,760). We can group our data by machine, date, and colour, which allows us to reduce our data set to no more than 16 records for each day. In the course of a year, that's 2 million source records summarized in less than 6,000!

Now we need an implementation – let's look at the options.

The hard way – do-it-yourself

The simplest way to achieve our summary table objective, in a simple environment, is to have the application maintain the summary. It sounds straightforward, but complexity grows exponentially, even if the logic is placed in triggers. It's also unnecessary because DB2 can do the work for us.

Immediate results

With DB2, the functionality we need to reduce the workload of our reporting is built right into the database, by utilizing Automatic Summary Tables (ASTs). An AST is a materialized view – a view that is stored on disk for direct access.

To use Automatic Summary Tables we simply:

1 Decide how to group the data, and whether we want it updated immediately (refresh immediate) or on-request (refresh deferred).

2 Create a summary table for the summary and optionally index it for even greater performance (see below).

Create a summary table:

```
CREATE SUMMARY TABLE TEST.DOM_PRODUCT_ST AS (
  SELECT Machine, Color, DateMade,
  SUM(Weight) AS SumWeight,
  COUNT(Weight) AS CountWeight,
  COUNT(*) AS Bags
  FROM TEST.DOM_PRODUCT
  GROUP BY CUBE(Machine, Color, DateMade)
) DATA INITIALLY DEFERRED REFRESH IMMEDIATE @
refresh table TEST.DOM_PRODUCT_ST @

create index TEST.DOMPRDSTMCIX ON  TEST.DOM_PRODUCT_ST
(Machine, Color) @
create index TEST.DOMPRDSTDMCIX ON  TEST.DOM_PRODUCT_ST
(DateMade, Machine, Color) @

-- Do a RUNSTATS, but not until you've populated the table!
RUNSTATS ON TABLE TEST.DOM_PRODUCT_ST WITH DISTRIBUTION
```

```
AND DETAILED INDEXES ALL @
RUNSTATS ON TABLE TEST.DOM_PRODUCT_ST WITH DISTRIBUTION
AND DETAILED INDEXES ALL @

-- DB2's query optimizer will make better-informed decisions
-- if you perform a RUNSTATS on both the source table
-- and the summary table while they are both populated
```

SELECT YOUR REFRESHMENT

You could now change your reports to query the summary table or you can continue using the original queries. First you need to decide whether to use *refresh immediate* or *refresh deferred* ASTs.

Refresh immediate summary tables:

- Are defined by:

```
CREATE SUMMARY TABLE summaryname
[(columname, columname,..)]
AS (fullselect)
DATA INITIALLY DEFERRED REFRESH IMMEDIATE
```

- Operate synchronously, that is, they are updated as part of any transaction on their source tables. This incurs some overhead, but the results are always available.
- Can be used by DB2 to automatically optimize both static and dynamic SQL queries.
- Must contain a GROUP BY clause and a COUNT(*) or COUNT_BIG(*).
- May use GROUPING SETS, CUBE, and ROLLUP (see listing below).
- May contain a SUM(columname) column function, but requires a COUNT(columname) on the same column.
- May be based on an inner-join between tables or views, but cannot use the INNER JOIN keywords.

In the listing below, we use a CUBE summary table with our revised queries to give us optimal performance. Query A can now be answered from a single row.

A CUBE summary table with revised queries:

```
CREATE SUMMARY TABLE TEST.DOM_PRODUCT_CUBE AS (
  SELECT Machine, Color, DateMade,
  SUM(Weight) AS SumWeight,
  COUNT(Weight) AS CountWeight,
  COUNT(*) AS Bags
  FROM TEST.DOM_PRODUCT  GROUP BY CUBE(Machine, Color, DateMade)
) DATA INITIALLY DEFERRED REFRESH IMMEDIATE @
refresh table TEST.DOM_PRODUCT_CUBE@
create index TEST.DOMPRDCDMCIX ON  TEST.DOM_PRODUCT_CUBE (
DateMade, Machine, Color) @
-- We'll only use one index this time, as
-- machine/colour totals can be found where DateMade is null
RUNSTATS ON TABLE TEST.DOM_PRODUCT_CUBE WITH DISTRIBUTION AND
DETAILED INDEXES ALL @
-- You can DROP TABLE TEST.DOM_PRODUCT_ST @
-- as DOM_PRODUCT_CUBE does the same plus more.
-- Now we can use queries on the original table,
-- and DB2 should choose the AST,
-- or we can explicitly query the cube AST as follows:

-- Query A
SELECT Machine,
  SumWeight/Bags as AveWeight
FROM TEST.DOM_PRODUCT_CUBE
WHERE Machine IS NOT NULL
  AND DateMade IS NULL
  AND Color IS NULL@

-- Query B
SELECT Color,
  SUM(SumWeight) as SumWeight,
  SUM(Bags) as Bags,
  SUM(SumWeight)/sum(Bags) as AveWeight
FROM TEST.DOM_PRODUCT_CUBE
WHERE DateMade BETWEEN (current date - 7 days)
  AND (current date - 1 day)
  AND Machine IS NOT NULL
  AND Color IS NOT NULL
GROUP BY Color@

-- Query C
-- Also includes an all-colours total
SELECT Color,
  Bags, SumWeight
FROM TEST.DOM_PRODUCT_CUBE
WHERE DateMade = CURRENT DATE
  AND Machine IS NULL
GROUP BY Color, Bags, SumWeight@
```

Refresh immediate summary tables are good when:

- Having a summary table will avoid the need to scan part or all of its source table.
- You can afford the transaction overhead.
- You need a completely up-to-date summary.
- Using a regular query or view is too slow, or fails due to other constraints.

UPDATE ON REQUEST

Examining our queries more carefully, we might:

- Decide that query A is acceptable without the current day's data.
- Recognize that query B doesn't need the current day's data.
- Establish that query C runs fast enough on the raw data.

Hence, we could skip the overhead of a refresh immediate summary, and rebuild the summary each day. A *refresh deferred* summary table is a neat way to do this. It allows more flexible queries, but has its own limitations.

Refresh deferred summary tables are similar to refresh immediate, except they:

- Operate asynchronously – they are refreshed only when you explicitly refresh them.
- May use scalar functions.
- May use other column functions, such as MIN and MAX.
- Do not require a COUNT(*) or COUNT_BIG(*).
- May contain a HAVING clause.
- Cannot be used to automatically optimize static SQL queries.

The listing below shows a *refresh deferred* definition. Remember,

you can increase flexibility and reduce overhead by deferring the update.

Create a deferred summary table:

```
CREATE SUMMARY TABLE TEST.DOM_PRODUCT_DEF AS (
  SELECT Machine, DateMade,
  SUM(Weight) AS SumWeight,
  COUNT(Weight) AS CountWeight,
  COUNT(*) AS Bags,
  MIN(SerialNo) as MinSerial,
  MAX(SerialNo) as MaxSerial
  FROM TEST.DOM_PRODUCT
  GROUP BY Machine, DateMade
) DATA INITIALLY DEFERRED REFRESH DEFERRED @

create index TEST.DOMPRDDEFMDIX ON  TEST.DOM_PRODUCT_DEF
(Machine, DateMade) @
```

Note that the contents of DOM_PRODUCT_DEF are updated only on the command refresh table TEST.DOM_PRODUCT_DEF, meaning that the contents of DOM_PRODUCT_DEF are not guaranteed to be current.

Refresh deferred summary tables are good when:

- You can refresh a summary table once and query it many times, because of the timing of the transactions and queries – for example if you run queries on historical daily summaries or the source data arrives in bursts.
- You need the extra functions that aren't available with refresh immediate.

So it's still a balancing act, and there's no single answer to all situations, but now you have two tools to cut your query times when collating large data sets. Decide whether your summary needs to be live, or simply refreshable, and let DB2 work smart.

Vikas Baruah
Senior Technical Specialist
American Management Systems (USA)

How to get DB2 entity-relationship diagrams via your Web browser – part 2

This month we conclude the code that will produce DB2 entity relationship diagrams from a Web browser.

```
      follow = current.next;
      while(follow != null) {
        nadjfol = 0;
        currentL = follow.theadjList.firstL;
        while(currentL != null) {
          nadjfol++;
          currentL = currentL.nextL;
        }
        if (nadjcur < nadjfol) {
          help = new Vertex("AAA", "BBB");
          help.tbcreator = current.tbcreator;
          help.tbname = current.tbname;
          help.theKey = current.theKey;
          help.theAtt = current.theAtt;
          help.theadjList = current.theadjList;
          current.tbcreator = follow.tbcreator;
          current.tbname = follow.tbname;
          current.theKey = follow.theKey;
          current.theAtt = follow.theAtt;
          current.theadjList = follow.theadjList;
          follow.tbcreator = help.tbcreator;
          follow.tbname = help.tbname;
          follow.theKey = help.theKey;
          follow.theAtt = help.theAtt;
          follow.theadjList = help.theadjList;
          nadjcur = nadjfol;
        }
        follow = follow.next;
      }
      current = current.next;
    }
    int widV = 0;
    int heiV = 0;
    int helpwidV;
    int helpheiV;
    int helpcolv;
    int i;
    int row = 0;
    int col = 0;

    current = first;
```

```
        while(current != null) {
          if (!current.visited) {
            row++;
            col = 1;
            current.rowv = row;
            current.colv = col;
            current.visited = true;
          }
          i = -1;
          currentL = current.theadjList.firstL;
          while(currentL != null) {
            if (!currentL.tbcreatorL.equals(current.tbcreator) ||
                !currentL.tbnameL.equals(current.tbname)) {
              follow = current;
              while(follow != null) {
                if (currentL.tbcreatorL.equals(follow.tbcreator) &&
                    currentL.tbnameL.equals(follow.tbname) &&
!follow.visited) {
                  i++;
                  follow.rowv = current.rowv + 1;
                  if (follow.rowv > row) row = follow.rowv;

                  help = first;
                  helpcolv = current.colv + i;
                  while(help != null) {
                 if (help.rowv == follow.rowv && help.colv >= helpcolv)
                      helpcolv = help.colv + 1;
                    help = help.next;
                  }
                  follow.colv = helpcolv;
                  follow.strong = false;
                  follow.visited = true;
                  break;
                 } else if (follow.visited) follow.strong = false;
                 follow = follow.next;
              }
            }
            currentL = currentL.nextL;
          }
          if (i > -1) {
            help = first;
            while(help != null) {
             if (help.rowv == current.rowv && help.colv > current.colv)
                help.colv += i;
              help = help.next;
            }
            current.colv += (i + 1) / 2;
          }
          current = current.next;
        }
```

```
          current = first;
          while(current != null) {
            helpwidV = widthVertex(current);
            helpheiV = heightVertex(current);
            if (helpwidV > widV) widV = helpwidV + 100;
            if (helpheiV > heiV) heiV = helpheiV + 100;
            current = current.next;
          }

          current = first;
          while(current != null) {
            current.xv = (current.colv - 1) * widV;
            current.yv = (current.rowv - 1) * heiV;
            helpwidV = widthVertex(current);
            helpheiV = heightVertex(current);
            current.dxv = current.xv + helpwidV;
            current.dyv = current.yv + helpheiV;
            if (current.dxv > maxdx) maxdx = current.dxv;
            if (current.dyv > maxdy) maxdy = current.dyv;
            current = current.next;
          }

        }
      }

      public int widthVertex(Vertex curr) {
        FontMetrics fontMetrics = getFontMetrics(font);
        String maxwidth = curr.tbcreator + "." + curr.tbname;
        DoublyLinkedListKey theKey;
        Key currentK;
        DoublyLinkedListOtherAttr theAtt;
        OtherAttr currentA;
        currentK = curr.theKey.firstK;
        while(currentK != null) {
          if ((currentK.name + " " + currentK.description +
               currentK.fklabel).length() > maxwidth.length()) {
             maxwidth = currentK.name + " " +
currentK.description + currentK.fklabel;
          }
          currentK = currentK.nextK;
        }
        currentA = curr.theAtt.firstA;
        while(currentA != null) {
          if ((currentA.nameA + " " + currentA.descriptionA +
               currentA.fklabelA).length() > maxwidth.length()) {
            maxwidth = currentA.nameA + " " +
currentA.descriptionA + currentA.fklabelA;
          }
          currentA = currentA.nextA;
        }
```

```
  return fontMetrics.stringWidth(maxwidth) + 4;
}

public int heightVertex(Vertex curr) {
  FontMetrics fontMetrics = getFontMetrics(font);
  DoublyLinkedListKey theKey;
  Key currentK;
  DoublyLinkedListOtherAttr theAtt;
  OtherAttr currentA;
  currentK = curr.theKey.firstK;
  int ncols = 0;
  int knum = 0;
  int anum = 0;
  while(currentK != null) {
    knum++;
    ncols++;
    currentK = currentK.nextK;
  }
  currentA = curr.theAtt.firstA;
  while(currentA != null) {
    anum++;
    ncols++;
    currentA = currentA.nextA;
  }
  if (knum == 0) {
    ncols = ncols + 3;
   }
  if (anum == 0) {
    ncols = ncols + 3;
   }
  return fontMetrics.getHeight() * ncols + 4;
}

class draw_area extends Canvas {
  Vertex current;
  Vertex follow;
  DoublyLinkedListKey theKey;
  Key currentK;
  DoublyLinkedListOtherAttr theAtt;
  OtherAttr currentA;
  DoublyLinkedListadjList theadjList;
  adjList currentL;
  Font font;

  protected draw_area(int mdx, int mdy, Font font) {
    super();
    this.font = font;
    this.setSize(mdx, mdy);
  }
```

```
public void paint(Graphics g) {
  FontMetrics fontMetrics = g.getFontMetrics(font);
  int rectWidth;
  int x;
  int y;
  int xx;
  int yy;
  int knum;
  int anum;
  int ncols;
  int i;
  int[] pos = new int[8];
  int offs = 10;
  int offs1 = 5;
  current = first;
  while(current != null) {
    x = current.xv + 50;
    y = current.yv + 50;
    xx = x;
    yy = y;
    rectWidth = (current.dxv - current.xv) + 4;
  g.drawString(current.tbcreator + "." + current.tbname, x, y);
    x = x + 2;
    knum = 0;
    anum = 0;
    ncols = 0;
    currentK = current.theKey.firstK;
    while(currentK != null) {
      knum++;
      ncols++;
      y = y + fontMetrics.getHeight();
      g.drawString(currentK.name + " " + currentK.description +
                   currentK.fklabel, x, y);
      currentK = currentK.nextK;
    }
    if (knum < 2) {
      y = y + fontMetrics.getHeight() * (2 - knum);
      ncols = ncols + 2 - knum;
    }
    y = y + fontMetrics.getDescent();
    g.drawLine(xx, y, xx + rectWidth, y);
    currentA = current.theAtt.firstA;
    while(currentA != null) {
      anum++;
      ncols++;
      y = y + fontMetrics.getHeight();
    g.drawString(currentA.nameA + " " + currentA.descriptionA +
                   currentA.fklabelA, x, y);
      currentA = currentA.nextA;
    }
```

```
            if (anum < 2) {
              y = y + fontMetrics.getHeight() * (2 - anum);
              ncols = ncols + 2 - anum;
            }
            int rectHeight = fontMetrics.getHeight() * ncols + 4;
      if (current.strong) g.drawRect(xx, yy + 2, rectWidth, rectHeight);
        else g.drawRoundRect(xx, yy + 2, rectWidth, rectHeight, 20, 20);
            current = current.next;
          }
          current = first;
          while(current != null) {
            for (i=0;i<8;i++) pos[i] = 0;
            currentL = current.theadjList.firstL;
            while(currentL != null) {
              if (currentL.tbcreatorL.equals(current.tbcreator) &&
                  currentL.tbnameL.equals(current.tbname)) {
                g.drawRect(current.dxv + 25, current.dyv + 68, 70, 40);
                g.fillOval(current.dxv + 48, current.dyv + 65, 10, 10);
g.drawString(currentL.deleteruleL, current.dxv + 60, current.dyv + 100);
              } else {
                follow = first;
                while(follow != null) {
                  if (currentL.tbcreatorL.equals(follow.tbcreator) &&
                      currentL.tbnameL.equals(follow.tbname)) {
                    if (current.xv == follow.xv) {
                      if (current.yv < follow.yv) {
                        pos[0] = pos[0] + 1;
                        x = (current.dxv + current.xv) / 2;
                        y = current.dyv;
                        xx = (follow.dxv + follow.xv) / 2;
                        yy = follow.yv;
                        x += pos[0] * offs;
                        xx += pos[0] * offs;
                        x -= offs1;
                        xx -= offs1;
                      } else {
                        pos[1] = pos[1] + 1;
                        x = (current.dxv + current.xv) / 2;
                        y = current.yv;
                        xx = (follow.dxv + follow.xv) / 2;
                        yy = follow.dyv;
                        x += pos[1] * offs;
                        xx += pos[1] * offs;
                        x -= offs1;
                        xx -= offs1;
                      }
                    } else {
                      if (current.yv == follow.yv) {
                        if (current.xv < follow.xv) {
                          pos[2] = pos[2] + 1;
```

```
      x = current.dxv;
      y = (current.dyv + current.yv) / 2;
      xx = follow.xv;
      yy = (follow.dyv + follow.yv) / 2;
      y += pos[2] * offs;
      yy += pos[2] * offs;
      y -= offs1;
      yy -= offs1;
    } else {
      pos[3] = pos[3] + 1;
      x = current.xv;
      y = (current.dyv + current.yv) / 2;
      xx = follow.dxv;
      yy = (follow.dyv + follow.yv) / 2;
      y += pos[3] * offs;
      yy += pos[3] * offs;
      y -= offs1;
      yy -= offs1;
    }
  } else {
    if (current.yv < follow.yv) {
      if (current.xv < follow.xv) {
        pos[4] = pos[4] + 1;
        x = (current.dxv + current.xv) / 2;
        y = current.dyv;
        xx = follow.xv;
        yy = (follow.dyv + follow.yv) / 2;
        x -= pos[4] * offs;
        yy += pos[4] * offs;
        x += offs1;
        yy -= offs1;
      } else {
        pos[5] = pos[5] + 1;
        x = (current.dxv + current.xv) / 2;
        y = current.dyv;
        xx = follow.dxv;
        yy = (follow.dyv + follow.yv) / 2;
        x += pos[5] * offs;
        yy += pos[5] * offs;
        x -= offs1;
        yy -= offs1;
      }
    } else {
      if (current.xv < follow.xv) {
        pos[6] = pos[6] + 1;
        x = (current.dxv + current.xv) / 2;
        y = current.yv;
        xx = follow.xv;
        yy = (follow.dyv + follow.yv) / 2;
        x += pos[6] * offs;
```

```
                        yy += pos[6] * offs;
                        x -= offs1;
                        yy -= offs1;
                      } else {
                        pos[7] = pos[7] + 1;
                        x = (current.dxv + current.xv) / 2;
                        y = current.yv;
                        xx = follow.dxv;
                        yy = (follow.dyv + follow.yv) / 2;
                        x -= pos[7] * offs;
                        yy += pos[7] * offs;
                        x += offs1;
                        yy -= offs1;
                      }
                    }
                  }
                }
     if (!currentL.strongL) g.drawLine(x+51, y+65, xx+51, yy+65);
                else {
                  for (i=0;i<2;i++) {
                    g.drawLine(x+51+i,y+65,xx+51+i,yy+65);
                  }
                }
                g.fillOval(xx+49,yy+60,10,10);
   g.drawString(currentL.deleteruleL, (x+xx+90)/2, (y+yy+120)/2);
                break;
              }
              follow = follow.next;
            }
          }
          currentL = currentL.nextL;
        }
        current = current.next;
      }
    }
  }
  public void draw() {
    maxdx = maxdx + 100;
    maxdy = maxdy + 100;
    Button printButton = new Button("Print");
    sp = new ScrollPane(ScrollPane.SCROLLBARS_AS_NEEDED);
    sp.setSize(size().width - 50, size().height - 1);
    da = new draw_area(maxdx, maxdy, font);
    sp.add(da);
    add(sp);
    add(printButton);
    printButton.addActionListener(new ActionListener(){
      public void actionPerformed(ActionEvent event){
        printComp(da);
      }
```

```
        });
      }
 // -- Key Manipulation---------------------------------------------
      public boolean isEmptyKey(DoublyLinkedListKey curr) {
        return curr.firstK==null;
      }
      public void insertLastKey(String kna, String kde) {
        Key newKey = new Key(kna, kde);
        if (isEmptyKey(last.theKey))
          last.theKey.firstK = newKey;
        else {
          last.theKey.lastK.nextK = newKey;
          newKey.previousK = last.theKey.lastK;
        }
        last.theKey.lastK = newKey;
      }
 // -- Other Attr Manipulation-------------------------------------
      public boolean isEmptyOtherAttr(DoublyLinkedListOtherAttr curr) {
        return curr.firstA==null;
      }
      public void insertLastOtherAttr(String ana, String ade) {
        OtherAttr newAtt = new OtherAttr(ana, ade);
        if (isEmptyOtherAttr(last.theAtt))
          last.theAtt.firstA = newAtt;
        else {
          last.theAtt.lastA.nextA = newAtt;
          newAtt.previousA = last.theAtt.lastA;
        }
        last.theAtt.lastA = newAtt;
      }
 // -- adjList Manipulation----------------------------------------
      public boolean isEmptyadjList(DoublyLinkedListadjList curr) {
        return curr.firstL==null;
      }
    public void insertLastadjList(Vertex curr, String tbcL, String tbnL,
                                  String delR, boolean ind) {
        adjList newadjList = new adjList(tbcL, tbnL, delR, ind);
        if (isEmptyadjList(curr.theadjList))
          curr.theadjList.firstL = newadjList;
        else {
          curr.theadjList.lastL.nextL = newadjList;
          newadjList.previousL = curr.theadjList.lastL;
        }
        curr.theadjList.lastL = newadjList;
      }
public void insertadjList(Connection con, ResultSet rs, Statement stmt,
 String filter) {
        Vertex current = first;
        String fk;
        int j;
```

```
          ResultSet rs2;
          Statement stmt2;
          DoublyLinkedListKey theKey;
          Key currentK;
          DoublyLinkedListOtherAttr theAtt;
          OtherAttr currentA;
          boolean ind;
          Vertex follow;
          while(current != null) {
            try {
              stmt = con.createStatement();
              rs = stmt.executeQuery("SELECT * " +
                     "FROM sysibm.sysrels " +
                "WHERE reftbcreator = '" + current.tbcreator + "' AND " +
                           "reftbname = '" + current.tbname + "' AND " +
                           "creator = '" + filter + "' " +
                     "ORDER BY CREATOR, TBNAME");
              while (rs.next()) {
                follow = first;
                while(follow != null) {
                  if
(follow.tbcreator.equals(rs.getString("CREATOR").trim()) &&
              follow.tbname.equals(rs.getString("TBNAME").trim())) break;
                  follow = follow.next;
                }
                ind = false;
                if (!pc) {                                        // host
                  stmt2 = con.createStatement();
                  rs2 = stmt2.executeQuery("SELECT * " +
                  "FROM sysibm.sysforeignkeys " +
         "WHERE creator = '" + rs.getString("CREATOR").trim() + "' AND " +
                  "tbname = '" + rs.getString("TBNAME").trim() + "' AND " +
                    "relname = '" + rs.getString("RELNAME").trim() + "' " +
                    "ORDER BY COLSEQ");
                    while (rs2.next()) {
                      fk = rs2.getString("COLNAME").trim();
                      currentK = follow.theKey.firstK;
                      while(currentK != null) {
                        if (currentK.name.equals(fk)) {
                          ind = true;
                          currentK.fklabel = " (fk)";
                          break;
                        }
                        currentK = currentK.nextK;
                      }
                      currentA = follow.theAtt.firstA;
                      while(currentA != null) {
                        if (currentA.nameA.equals(fk)) {
                          currentA.fklabelA = " (fk)";
                          break;
```

```
                }
                currentA = currentA.nextA;
              }
            }
            rs2.close();
            stmt2.close();
          } else {                                          // pc
            for (j=0; j < rs.getShort("COLCOUNT"); j++) {
    fk = rs.getString("FKCOLNAMES").substring((j * 2Ø) + 1,2Ø).trim();
              currentK = follow.theKey.firstK;
              while(currentK != null) {
                if (currentK.name.equals(fk)) {
                  ind = true;
                  currentK.fklabel = " (fk)";
                  break;
                }
                currentK = currentK.nextK;
              }
              currentA = follow.theAtt.firstA;
              while(currentA != null) {
                if (currentA.nameA.equals(fk)) {
                  currentA.fklabelA = " (fk)";
                  break;
                }
                currentA = currentA.nextA;
              }
            }
          }
          insertLastadjList(current, rs.getString("CREATOR").trim(),
                            rs.getString("TBNAME").trim(),
                            rs.getString("DELETERULE").trim(),
                            ind);
        }
        rs.close();
        stmt.close();
      } catch(SQLException ex) {
        setError("SQLException: " + ex);
      }
      current = current.next;
    }
  }
}
// ------------------------------------------------------------
  public void init() {
    font = getFont();
    queryResults.addElement("OK");
    String vtbcreator = "";
    String vtbname = "";
    String vstr;
    try {
```

```
        Class.forName("COM.ibm.db2.jdbc.net.DB2Driver").newInstance();
      } catch (Exception ex) {
        setError("Can't find Database driver class: " + ex);
        return;
      }
      DoublyLinkedListVertex theList = new DoublyLinkedListVertex();
      try {
        Connection con = null;
        String server = getParameter("server");
        String port = getParameter("port");
        String dbname = getParameter("dbname");
        String url = "jdbc:db2://"+server+":"+port+"/"+dbname;
        String userid = getParameter("userid");
        String passwd = getParameter("password");
        String filter = getParameter("creator");
        con = DriverManager.getConnection(url, userid, passwd);
        Statement stmt1 = con.createStatement();
        ResultSet rs1 = stmt1.executeQuery("SELECT * " +
                                           "FROM sysibm.systables " +
                                         "WHERE creator = 'SYSIBM' AND " +
                                               "name = 'SYSFOREIGNKEYS'");
        if (rs1.next()) {                                       // host
          pc = false;
        } else {                                                // pc
          pc = true;
        }
        rs1.close();
        stmt1.close();
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT a.* " +
                       "FROM sysibm.syscolumns a, sysibm.systables b " +
                       "WHERE a.tbcreator = '" + filter + "' and " +
                             "a.tbcreator = b.creator and " +
                             "a.tbname = b.name and " +
                             "b.type = 'T' " +
                       "ORDER BY a.TBCREATOR, a.TBNAME, a.COLNO");
        while (rs.next()) {
          if (!rs.getString(3).trim().equals(vtbcreator) ||
              !rs.getString(2).trim().equals(vtbname)) {
            vtbcreator = rs.getString(3).trim();
            vtbname = rs.getString(2).trim();
            theList.insertLastVertex(rs.getString(3).trim(),
rs.getString(2).trim());
          }
          vstr = rs.getString("COLTYPE").trim();
          if (vstr.equals("CHAR") | vstr.equals("VARCHAR")) {
            vstr = vstr + "(" + rs.getShort("LENGTH") + ")";
          } else {
            if (vstr.equals("DECIMAL")) {
              vstr = vstr + "(" + rs.getShort("LENGTH") + "," +
```

```
rs.getShort("SCALE") + ")";
          } else {
            if (vstr.equals("BLOB") | vstr.equals("CLOB")) {
              if (pc) {
                vstr = vstr + "(" + rs.getInt("LONGLENGTH") + ")";
              } else {
                vstr = vstr + "(" + rs.getInt("LENGTH2") + ")";
              }
            }
          }
        }
        if (rs.getString("NULLS").trim().equals("N")) {
          vstr = vstr + " NOT NULL";
        }
        if (rs.getShort("KEYSEQ") > Ø) {
          theList.insertLastKey(rs.getString(1).trim(), vstr);
        } else {
          theList.insertLastOtherAttr(rs.getString(1).trim(), vstr);
        }
      }
      rs.close();
      stmt.close();
      theList.insertadjList(con, rs, stmt, filter);
      con.close();
      theList.arrangeVertex();
      theList.draw();
    } catch(SQLException ex) {
      setError("SQLException: " + ex);
    }
  }
  public void paint(Graphics g) {
    if (queryResults == null) {
      g.drawString(message, 5, 5Ø);
      return;
    }
  }
  private void setError(String mess) {
    queryResults = null;
    message = mess;
    repaint();
  }
  private Frame getFrame(Component c) {
    while((c = c.getParent())!=null) {
      if(c instanceof Frame) return (Frame) c;
    }
    return null;
  }
  private void printComp(Component c) {
    PrintJob pjob = getToolkit().getPrintJob(getFrame(c),
"ERDB2", (Properties)null);
```

```
      if (pjob != null) {
        Graphics pg = pjob.getGraphics();
        if (pg != null) {
          sp.setScrollPosition(0, 0);
          Point p = sp.getScrollPosition();
          Dimension vp = sp.getViewportSize();
          Dimension size = c.getSize();
          Dimension pagesize = pjob.getPageDimension();
          for (int i=0; i < (size.height/vp.height)+1; i++) {
            for (int j=0; j < (size.width/vp.width)+1; j++) {
              printAll(pg);
              pg.dispose();
              p.x = p.x + vp.width + 1;
              sp.setScrollPosition(p.x, p.y);
              pg = pjob.getGraphics();
            }
            p.x = 0;
            p.y = p.y + vp.height + 1;
            sp.setScrollPosition(p.x, p.y);
          }
          pg.dispose();
        }
        pjob.end();
        sp.setScrollPosition(0, 0);
      }
    }
  }
```

Nikola Lazovic
DB2 System Administrator
Postal Savings Bank (Yugoslavia)

DB2 news

IBM has announced DB2 Content Manager Version 8.2, which provides enhanced integration capabilities across IBM's e-infrastructure middleware, including DB2 UDB, WebSphere, Lotus, and Tivoli. The new software enables customers to more easily search and retrieve critical business information, replicate information twice as fast, and more easily publish information to the Web for increased usage across the enterprise.

IBM also announced DB2 Records Manager Version 2.1m, which enables organizations to embed record keeping policies into their existing business applications, making it easier to declare and manage business documents as e-records. Now administrators can more easily embed records management capabilities into their existing solutions and apply records management capabilities to physical content in addition to electronic information.

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

* * *

Embarcadero has announced Versions 7.1 of DBArtisan and Rapid SQL, including enhanced database platform support with particular focus on DB2 Universal Database and new support for OS/390.

DBArtisan is for managing DB2, Oracle, Sybase, and Microsoft SQL Server, enabling administrators to concurrently manage multiple databases from a single graphical console.

Rapid SQL is an integrated development environment that enables developers to create, edit, version, tune, and deploy server-side objects residing on DB2, Microsoft SQL Server, Oracle, and Sybase databases.

With extended support in the new versions for DB2 UDB Enterprise-Extended Edition (EEE), DBAs and database developers don't have to leave the graphical consoles of DBArtisan or Rapid SQL to administer or develop applications in their partitioned environments.

Version 7.1 provides increased support for OS/390, including a new interface for cross-referencing DBRMs and plans and packages, along with associated SQL statements.

New features of DBArtisan 7.1 include extended support for DB2 UDB-DB 7.2 EEE and application processes support for DB2 OS/390, covering DBRMs, cross-reference associated plans and packages, view and explain package statements, and bind, rebind, and free plans and packages.

In Rapid SQL 7.1, there's now support for UDB 7.2 EEE, and standard support for DB2 OS/390, including project management (including integration with SCC compliant Version Control Systems), code generation for statements and procedures, OS/390 favourite code templates, and paste SQL and SQL Syntax.

For further information contact:
Embarcadero, 425 Market Street, Suite 425, San Francisco, CA 94105, USA.
Tel: (415) 834 3131.
URL: http://www.embarcadero.com/news/DBArtisan_Rapid71.asp.

* * *