152

DB2 update

June 2005

In this issue

A tc PUBLICATION

DB2 Update

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

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

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Colin Smith
E-mail: info@xephon.com

Subscriptions and back-issues

A year's subscription to *DB2 Update*, comprising twelve monthly issues, costs $380.00 in the USA and Canada; £255.00 in the UK; £261.00 in Europe; £267.00 in Australasia and Japan; and £265.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 2000 issue, are available separately to subscribers for $33.75 (£22.50) each including postage.

DB2 Update on-line

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

Contributions

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

Some SQL tricks for the DB2 developer

It is always a good idea to keep your bag of SQL tricks loaded with techniques to help solve some of the more common or troubling application development problems. I do not know if the following SQL techniques match up to your most pressing development issues, but I'm sure they will come in handy in certain situations.

SORTING DAYS OF THE WEEK

Here is a sorting trick that you can use when you are dealing with temporal data. Assume that you have a table containing transactions. The table has a CHAR(3) column containing the name of the day on which the transaction happened; let's call this column DAY_NAME. Now, let's further assume that we want to write queries against this table that orders the results by DAY_NAME. We'd want Sunday first, followed by Monday, Tuesday, Wednesday, and so on. How can this be done?

Well, if we write the first query that comes to mind, the results will obviously be sorted improperly:

```
SELECT   DAY_NAME, COL1, COL2 . . .
FROM     TXN_TABLE
ORDER BY DAY_NAME;
```

The results from this query would be ordered alphabetically; in other words:

```
FRI
MON
SAT
SUN
THU
TUE
WED
```

One solution would be to design the table with an additional numeric or alphabetic column that would sort properly. By this I mean we could add a DAY_NUM column that would be 1 for

Sunday, 2 for Monday, and so on. But this requires a database design change, and it becomes possible for the DAY_NUM and DAY_NAME to get out of sync.

A better solution uses just SQL and requires no change to the database structures. All you need is an understanding of SQL and SQL functions – in this case, the LOCATE function. Here is the SQL:

```
SELECT   DAY_NAME, COL1, COL2 . . .
FROM     TXN_TABLE
ORDER BY LOCATE('SUNMONTUEWEDTHUFRISAT',DAY_NAME);
```

The trick here is to understand how the LOCATE function works: it returns the starting position of the first occurrence of one string within another string. So, in our example, LOCATE finds the position of the DAY_NAME value within the string 'SUNMONTUEWEDTHUFRISAT', and returns the integer value of that position. If DAY_NAME is WED, the LOCATION function in the above SQL statement returns 10. (Note: some other database systems have a similar function called INSTR.) Sunday would return 1, Monday 4, Tuesday 7, Wednesday 10, Thursday 13, Friday 16, and Saturday 19. This means that our results would be in the order we require.

Of course, you can go one step further if you like. Some queries may need to actually return the day of week. You can use the same technique with a twist to return the day of week value given only the day's name. To turn this into the appropriate day of the week number (that is, a value of 1, 2, 3, 4, 5, 6 or 7), we divide by3, use the INT function on the result to return only the integer portion of the result, and then add 1:

```
INT(LOCATE('SUNMONTUEWEDTHUFRISAT',DAY_NAME)/3) + 1;
```

Let's use our previous example of Wednesday again. The LOCATION function returns the value 10. So, INT(10/3) = 3 and add 1 to get 4. And sure enough, Wednesday is the fourth day of the week.

REMOVING SUPERFLUOUS SPACES

We can all relate to dealing with systems that have data integrity problems. But some data integrity problems can be cleaned up using a touch of SQL. Consider the common data entry problem of extraneous spaces (or blanks) inserted into a name field. Not only is it annoying, sometimes it can cause the system to ignore relationships between data elements because the names do not match. For example, ‘Craig Mullins’ is not equivalent to ‘Craig Mullins’ – the first one has two spaces between the two parts of the name, whereas the second has only one.

You can write an SQL UPDATE statement to clean up the problem if you know how to use the REPLACE function. REPLACE does what it sounds like it would do: it reviews a source string and replaces all occurrences of one string with another. For example, to replace all occurrences of Z with A in the string BZNZNZ you would code:

```
REPLACE('BZNZNZ','Z','A')
```

The result would be BANANA. So, let’s create a SQL statement using REPLACE to get rid of any unwanted spaces in the NAME column of our EMPLOYEE table:

```
UPDATE EMPLOYEE
   SET NAME = REPLACE(
               REPLACE(
                REPLACE(NAME, SPACE(1), '<>')
                '><', SPACE(Ø))
               '<>', SPACE(1));
```

“Wait-a-minute,” you might be saying, “What are all of those left and right carats [greater than/less than signs] and why do I need them?” Well, let’s go from the inside out. The inside REPLACE statement takes the NAME column and converts every occurrence of a single space into a left/right carat. The next REPLACE (working outward), takes the string we just created, and removes every occurrence of a right/left carat combination by replacing it with a zero length string. The final REPLACE function takes that string and replaces any left/right carats with a single space. The reversal of the carats is the key to removing all spaces except one – remember, we want to retain a single space anywhere that there was aleady a single space as well as anywhere that there were multiple spaces. Try it; it works.

Of course, you can use any two characters you like, but the left and right carat characters work well visually. Be sure that you do not choose to use characters that occur naturally in the string that you are acting on.

Finally, the SPACE function was used for clarity. You could have used strings encased in single quotes, but the SPACE function is easier to read. It simply returns a string of spaces, the length of which is specified as the integer argument. So, SPACE(12) would return a string of 12 spaces.

AGGREGATING AGGREGATES

The final SQL trick we'll uncover in this article allows us to perform aggregations of aggregates. For example, you might want to compute the average of a sum. This comes up frequently in applications that are built around sales amounts. Let's assume that we have a table containing sales information. Each sales amount has additional information indicating the salesman, region, district, product, date, etc.

A common requirement is to produce a report of the average sales by region for a particular period, say the first quarter of 2005. But the data in the table is at a detail level, meaning that we have a row for each specific sale.

A novice SQL coder might try to write a query with a function inside a function, like AVG(SUM(SALE_AMT)). Of course, this is invalid SQL syntax. DB2 will not permit the nesting of aggregate functions. But we can use nested table expressions and our knowledge of SQL functions to build the correct query.

Let's start by creating a query to return the sum of all sales by region for the time period in question. That query should look something like this:

```
SELECT REGION, SUM(SALE_AMT)
FROM   SALES
WHERE SALE_DATE BETWEEN DATE('2005-01-01')
```

```
                AND     DATE('2005-03-31')
GROUP BY REGION;
```

Now that we have the total sales by region for the period in question, we can embed this query into a nested table expression in another query like so:

```
SELECT NTE.REGION, AVG(NTE.TOTAL_SALES)
FROM (SELECT REGION, SUM(SALE_AMT)
      FROM   SALES
      WHERE SALE_DATE BETWEEN DATE('2005-01-01')
                      AND     DATE('2005-03-31')
      GROUP BY REGION) AS NTE)
GROUP BY NTE.REGION;
```

SUMMARY

In this article we examined several techniques to solve application problems using only SQL. With a sound understanding of SQL, and particularly SQL functions and expressions, you can often find novel ways to solve thorny problems using nothing but SQL.

Craig S Mullins
Data Management Strategist
Mullins & Associates (USA)

Understanding Database Managed Space (DMS) table space maps and striping

Whether or not they are new to the IBM DB2 Universal Database for Linux, Unix, and Windows (DB2 UDB) product, many DBAs don't understand the concept of table space striping. Perhaps the main reason is that up until the DB2 UDB Version 8.1 release, it really wasn't a concept or object that was exposed to end users. This changed in the Version 8.1 release with the ability to avoid a rebalance (you'll learn more about this later). In this article, I'll introduce you to the concept of table space striping and how it works in the DB2 UDB product. This is important to understand in order to add space to your environment without affecting performance, or to leverage an up-and-coming new automated storage management feature in a future release of DB2 UDB.

A LITTLE BIT ABOUT DMS TABLE SPACES

In this article, I will assume that you know what a DMS table space is. However, it’s worth talking about some of the objects that are relevant to this article and used by DMS table spaces.

Each DMS table space has a meta-data structure we call a table space map. DB2 UDB uses this map to maintain a description of the layout of the table space containers and identify the location on the physical disk in which the extent of data persists. More specifically, the table space map defines the table space, the pool page number, the byte offset, and container, where data is written to disk. The map for a table space is stored in the table space’s SQLSPSC files.

A DMS table space is made up of one or more containers. When a DMS table space has multiple containers (they don’t have to be the same size), DB2 UDB stripes the data across the containers in order to achieve an even distribution of data

```
                        Containers

                      0       1       2
                   +-----------------
                 0|   0       1       2 }
        Stripes  1|   3       4       5 }
                 2|   6       7       8 }    Range 0
                 3|   9      10      11 }
                 4|  12      13         }
                 5|  14      15         }    Range 1
```

Figure 1: A table space map with three containers

across the storage. This helps to drive parallelism and, in the end, performance.

Think about this statement for a moment. If all of your data was on one disk, you’d possibly have one disk controller retrieving data from it. If you could spread that data evenly across three disks, you’d have three controllers retrieving data, three caches holding some of this data in high-speed access memory, three modes of redundancy, and more. By the way, for DBAs trying to achieve optimal performance, we recommend spreading the data across multiple disks.

For example, Figure 1 shows a table space map with three containers.

You could create the table space in Figure 1 using the following command:

```
CREATE TABLESPACE MYTS MANAGED BY DATABASE
 USING (FILE 'TS1' 75, FILE 'TS2' 75 FILE 'TS3' 55)
      EXTENTSIZE 1Ø
```

In Figure 1, each container is shown as a vertical bar of extents. Note that the two larger containers have six extents (0,1,2,3,4,5), while the third container has only four extents (0,1,2,4). Each horizontal line through the table space map is called a stripe.

When stripes all line up (for example, stripes 0,1,2,3) they are part of the same range. When a container is no longer part of a stripe, a new range is implicitly created (for example, when container TS3 becomes full, a new range is created).

You can see in the previous Figure and statement that there are three containers, in the MYTS table space. Note that container TS3 is smaller (55 extents versus 75 extents) than the TS1 and TS2 containers. (Container numbers in DB2 UDB start at 0, so these containers are listed as container 0,1, and 2 respectively – a shown in Figure 1.)

As you insert data into your table, DB2 UDB will stripe the data across extents that span all your table space’s containers in a round-robin fashion. In other words, once an extent is filled on container 0, another is filled on container 1, then on container 2, then back to container 0, and so on.

BEFORE DB2 UDB VERSION 8.1

The reason why many DBAs are not aware of the table space map is that before DB2 UDB V8.1 it was an internal-use only structure. In fact, the only way you could see the table space map in DB2 UDB V7 was by dumping the contents of the SQLSPCS files using the db2dart utility with the /DTSF option – something you'd probably only do with the assistance of DB2 support personnel.

In DB2 UDB V7, when you added a container to your table space, the data was automatically rebalanced across all the containers to achieve an even data distribution (more on this in the next section).

On the one hand, this was a good thing because the data would be balanced and this should mean better performance because of better parallelism, right? Well, sort of; in the long run, that's correct. However, to perform the rebalance, DB2 UDB would have to physically move the data from one container (disk) to another (I'll show you an example in the next section). Although this I/O operation is online, it could represent a slowdown to your system because a number of disk reads and writes could be occurring while end users are trying to get at the very data being rebalanced – they won't be able to access the data while it is in the process of being moved.

DB2 UDB VERSION 8.1 AND HOW TABLE SPACES WORK

DB2 UDB V8.1 introduced an option to force a new stripe set when adding a container to a table space. This gives DBAs the ability to avoid the performance hit of a rebalance (which, by the way, can be throttled as of DB2 V8.1.2 as well, but that's

```
                   Containers

              0       1       2       3
           +-----------------------
          0|  0       1       2       3 }
 Stripes  1|  4       5       6       7 }
          2|  8       9      10      11 }     Range 0
          3| 12      13      14      15 }
          4| 16      17              18 }
          5| 19      20              21 }     Range 1
```

Figure 2: MYTS table space map with a fourth container

another article) yet still leverage the new storage for new data inserts.

Because of this change, the table space map isn’t a ‘secret’ object any more. In fact, you can retrieve a table space map using a database snapshot that you can perform from a command prompt using a simple SQL statement.

To see the effects of a rebalance, consider adding a fourth container to our working example using the following statement:

```
    ALTER TABLESPACE MYTS ADD (FILE 'TS4' 75)
```

Now the table space map for MYTS might look like Figure 2.

Compare this new table space map to the previous one. You can see that extents 0 and 2 are in the same location in both maps. However, the rest of the extents are assigned to different physical positions – this is called a rebalance.

In general, any container operation that results in the existing location changing will result in a rebalance. As previously mentioned, depending on the amount of data to be moved, this could pose a performance issue. It follows that if all table space containers are the same size, you can grow them and avoid a rebalance altogether, and, quite honestly, that’s not a

```
                 Containers

              0     1     2     3     4     5
            +---------------------------------
           0|  0     1     2     3             }
Stripes    1|  4     5     6     7             } Range 0
           2|  8     9    10    11             }
           3| 12    13    14    15             }
           4| 16    17          18             }
           5| 19    20          21             } Range 1
           6|                         22    23 }
           7|                         24    25 } Range 2
           8|                         26    27 }
```

Figure 3: MYTS table space map with new stripe set

bad rule of thumb to follow – always try to KISS it (Keep It Simple Silly).

In DB2 UDB Version 8, the concept of a stripe set has been introduced to the DB2 UDB command syntax. This allows DBAs to add containers to their space-starved table spaces with a guarantee that no data will need to be rebalanced – thus providing better response times for transaction-heavy environments.

For example, the following command would add more containers to our working table space that we extended (which subsequently forced a rebalance), but avoid the rebalance taking place by leveraging the NEW STRIP SET option:

```
ALTER TABLESPACE MYTS BEGIN NEW STRIP SET
  (FILE 'TS5' 55, FILE 'TS6' 55)
```

After running this command, our table space map now looks like Figure 3. You can compare this table space map with Figure 2 and see that the physical locations of the extents never changed – thereby suggesting that a rebalance wasn't necessary.

Table space maps aren't difficult to understand, but they may be new to you. If you spend a couple of minutes ensuring that you understand the concepts laid out in this article, you should have no problems leveraging the enhancements in DB2 UDB Version 8.1 that allow you to grow your database without jeopardizing your service level agreements (SLA).

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

Displaying BUFFERPOOL attributes from BSDS data

This TSO REXX EXEC effectively takes the place of the COBOL II code originally published in *DB2 Update* in July 1995 that addressed the display of buffer pool attributes. BUFFERPOOL attributes (eg memory allocations, thresholds, etc) can be gathered from the X'0A000001' key record of a DB2 subsystem's BSDS dataset. The purpose of the EXEC is to return a quickly summarized screen shot of all parameter settings for defined buffer pools. The EXEC documents its processes. Please note the EXEC is suited for use with up to DB2 V7.1 at the moment.

The REXX source code is self-documenting.

DB2BUFFR

```
/*  Rexx -----------------------------------------------------------*/
/* Display buffer pool attribute information from BSDS record key    */
/* x'ØAØØØØØ1'.                                                      */
/* format to ivoke the EXEC: "TSO DB2BUFFR ssid"                     */
/* where 'ssid' = DB2T or DB2P or DB2L, etc.                         */
/*-----------------------------------------------------------------*/
/* trace ?r */
/*-----------------------------------------------------------------*/
/* buffer pools                                                      */
/*   5Ø  4K buffer pools = BPØ......BP49                             */
/*   1Ø  8K buffer pools = BP8KØ....BP8K9                            */
/*   1Ø 16K buffer pools = BP16KØ...BP16K9                           */
/*   1Ø 32K buffer pools = BP32K...BP32K9                            */
/*-----------------------------------------------------------------*/
parse upper arg parm
bsds_node1 = substr(parm,1,4)

/*-----------------------------------------------------------------*/
/* if no parm was passed (or if it's invalid), then default to db2z  */
/*-----------------------------------------------------------------*/
select
   when bsds_node1 = 'DB2T' then
        nop
   when bsds_node1 = 'DB2P' then
        nop
   when bsds_node1 = 'DB2L' then
        nop
   when bsds_node1 = 'DB2X' then
        nop
   when bsds_node1 = 'DB2Z' then
        nop
   otherwise
        bsds_node1 = 'DB2Z'
end

bsds_node2 = '.BSDSØ1'
bsds_dsn   = bsds_node1||bsds_node2

/* "ISPEXEC LIBDEF ISPPLIB DATASET ID('SDC.OPLIB.DB2X.DSNSPFP')"      */
/* "ISPEXEC LIBDEF SYSEXEC DATASET ID('SDC.OPLIB.DB2X.SYSEXEC')"      */
ADDRESS tso
/*------------------------------------------------------------------*/
/* Allocate the "input" BSDSØ1 (KSDS boot strap dataset).           */
/* Allocate the "output" TEMPØ1 temporary sequential dataset.       */
/*------------------------------------------------------------------*/
"ALLOC FI(BSDSØ1) DSN('"bsds_dsn"') SHR"
if rc ¬= Ø then do
   say 'error allocating BSDSØ1, RC=' rc
   exit rc
end
/*"ALLOC DD(TEMPØ1) NEW DELETE DSN('SDCTSO.TSSWJNM.TEMPØ1')"          */
"ALLOC DD(TEMPØ1) NEW DELETE
 LRECL(4Ø95) RECFM(V B) DSORG(PS)"
if rc ¬= Ø then do
   say 'error allocating TEMPØ1, RC=' rc
   exit rc
end

/*------------------------------------------------------------------*/
/* Trap the command messages from the IDCAMS REPRO service so that   */
/* they are not displayed on the terminal.                          */
/* Only display the trapped messages if the REPRO has an error.      */
/*------------------------------------------------------------------*/
x=OUTTRAP('idc_msg.','*','NOCONCAT')

/*------------------------------------------------------------------*/
/* Use the REPRO command to copy the x'ØAØØØØØ1' record into a       */
/* temporary file.                                                  */
/*------------------------------------------------------------------*/
"REPRO INFILE(BSDSØ1) OFILE(TEMPØ1) FROMKEY(X'ØAØØØØØ1') COUNT(1)"
if rc ¬= Ø then do
   say 'error REPROing the x'ØAØØØØØ1' record, RC=' rc
   do i = 1 to idc_msg.Ø
      say msg.i
   end
   exit rc
end

"FREE DD(BSDSØ1)"
x=OUTTRAP('OFF')
/*-----------------------------------------------------------------*/
/* Use EXECIO to retrieve the VSAM data from the temp dataset.       */
/*-----------------------------------------------------------------*/
"EXECIO 1 DISKR TEMPØ1"
if rc ¬= Ø then do
   say 'error reading the dataset, RC=' rc
   "FREE DD(TEMPØ1)"
   exit rc
end
PULL record
"EXECIO Ø DISKR TEMPØ1 (FINIS"
if rc ¬= Ø then do
   say 'error closing temp dataset, RC=' rc
   "FREE DD(TEMPØ1)"
   exit rc
end
"FREE DD(TEMPØ1)"

ADDRESS "ISPEXEC" "TBCREATE BUFLIST"||,
                  " KEYS(P1 P2 P3 P4 P5 P6 P7 P8 P9 P1Ø P11 P12)"||,
                  " NOWRITE REPLACE"

if rc ¬= Ø then do
   say 'error doing TBCREATE, RC=' rc
   exit rc
end

/*-----------------------------------------------------------------*/
/* Main-line processing.                                             */
/*                                                                   */
/* The buffer pool information starts at position x'39' (decimal 57) */
/* in record key (x'ØAØØØØØ1').  Each buffer has a 34 byte area      */
/* that describes its attributes; this repeats in consecutive        */
/* fashion from BPØØ to BPxx (eg there are 8Ø buffers).              */
/* The beginning byte of the buffer BPØ description begins in        */
/* position 57 of the record.                                        */
/*-----------------------------------------------------------------*/
positio = 57

do 8Ø
   call FORMAT_BSDS_OUT
end

PØ = bsds_node1 /* set the subsystem id into the PANEL */
```

```
ADDRESS "ISPEXEC" "TBTOP BUFLIST"
if rc ¬= Ø then do
   say 'error doing TBTOP, RC=' rc
   exit rc
end

ADDRESS "ISPEXEC" "TBDISPL BUFLIST PANEL(DB2BUFFR)"
select
   when rc = Ø then
      nop
   when rc = 8 then do /* the user entered return */
      ADDRESS "ISPEXEC" "TBEND BUFLIST"
      exit
      end
   otherwise do
      say 'error doing TBDISPL, RC=' rc
      ADDRESS "ISPEXEC" "TBEND BUFLIST"
      exit rc
      end
end

ADDRESS "ISPEXEC" "TBEND BUFLIST"
if rc ¬= Ø then do
   say 'error doing TBEND, RC=' rc
   exit rc
end
return /* we're done with the EXEC; let's leave */
/*-------------------------------------------------------------------*/
/* If the virtual pool is greater than zero or if the hiperpool is   */
/* greater than zero, then the buffer pool has potential for use.    */
/* Otherwise it is not being used, and there is no sense in          */
/* displaying information for a pool that is not used.               */
/* Add 32 to the displacement position to get to the start of the    */
/* next buffer's information.                                        */
/*-------------------------------------------------------------------*/
FORMAT_BSDS_OUT:

select
   when c2d(substr(record,positio+8,4))  > Ø  then /* is vpool > Ø?  */
        call DISCERN_FIELDS
   when c2d(substr(record,positio+12,4)) > Ø  then /* is hpool > Ø?  */
        call DISCERN_FIELDS
   otherwise nop
end

positio  = positio + 32

return
/*-------------------------------------------------------------------*/
/* Determine the flags for vtype, castout, pgsteal values, and all   */
/* other buffer information fields.                                   */
/*--------------------------------------------------------------------*/
/* In the first byte of each buffer information area, there are      */
/* flags that can be set for vtype, castout, and pgsteal.             */
/*                                                                    */
/* bit 8 is the flag for castout; bit 5 is the flag for vtype, and   */
/* bit 4 is the flag for pgsteal.                                     */
/*--------------------------------------------------------------------*/
DISCERN_FIELDS:
select
   when substr(record,positio,1) = 'ØØØØØØØØ'b then
        do
        vptype =  'P'
        castout = 'Y'
        pgsteal = 'L'
        end
   when substr(record,positio,1) = '1ØØØØØØØ'b then
        do
        vptype =  'P'
        castout = 'N'
        pgsteal = 'L'
        end
   when substr(record,positio,1) = '1ØØ1ØØØØ'b then
        do
        vptype =  'D'
        castout = 'N'
        pgsteal = 'L'
        end
   when substr(record,positio,1) = '1ØØ11ØØØ'b then
        do
        vptype =  'D'
        castout = 'N'
        pgsteal = 'F'
        end
   when substr(record,positio,1) = 'ØØØ11ØØØ'b then
        do
        vptype =  'D'
        castout = 'Y'
        pgsteal = 'F'
        end
   when substr(record,positio,1) = 'ØØØ1ØØØØ'b then
        do
        vptype =  'D'
        castout = 'Y'
        pgsteal = 'L'
        end
   when substr(record,positio,1) = 'ØØØØ1ØØØ'b then
        do
        vptype =  'P'
        castout = 'Y'
        pgsteal = 'F'
        end
   when substr(record,positio,1) = '10001000'b then
        do
        vptype =  'P'
        castout = 'N'
        pgsteal = 'F'
        end
   otherwise
        do
        vptype =  '?'
        castout = '?'
        pgsteal = '?'
        end
end

bpool    = c2d(substr(record,positio+3,1))
vpseqt   = c2d(substr(record,positio+4,1))
hpseqt   = c2d(substr(record,positio+5,1))
dwqt     = c2d(substr(record,positio+6,1))
vdwqt    = c2d(substr(record,positio+7,1))
vpsize   = c2d(substr(record,positio+8,4))
hpsize   = c2d(substr(record,positio+12,4))
vppseqt  = c2d(substr(record,positio+16,1))
vpxpseqt = c2d(substr(record,positio+17,1))

/*---------------------------------------------------------------------*/
/* BP0.....BP49   - in x'0A000001' record is value x'00' thru x'31'  */
/*                                              dec'00'       '49'  */
/* BP32K...BP32K9 - in x'0A000001' record is value x'50' thru x'59'  */
/*                                              dec'80'       '89'  */
/* BP8K0...BP8K9  - in x'0A000001' record is value x'64' thru  '6D'  */
/*                                              dec'100'      '109' */
/* BP16K0..BP16K9 - in x'0A000001' record is value x'78' thru x'81'  */
/*                                              dec'120'      '129' */
/*---------------------------------------------------------------------*/
select
   when bpool < 50  then
      nop
   when bpool < 90  then
      select
         when bpool = 80 then
              bpool = value('32K')
         when bpool = 81 then
              bpool = value('32K1')
         when bpool = 82 then
              bpool = value('32K2')
         when bpool = 83 then
              bpool = value('32K3')
         when bpool = 84 then
```

```
                bpool = value('32K4')
          when bpool = 85 then
                bpool = value('32K5')
          when bpool = 86 then
                bpool = value('32K6')
          when bpool = 87 then
                bpool = value('32K7')
          when bpool = 88 then
                bpool = value('32K8')
          when bpool = 89 then
                bpool = value('32K9')
          otherwise
             nop
       end
    when bpool < 110 then
       select
          when bpool = 100 then
                bpool = value('8K0')
          when bpool = 101 then
                bpool = value('8K1')
          when bpool = 102 then
                bpool = value('8K2')
          when bpool = 103 then
                bpool = value('8K3')
          when bpool = 104 then
                bpool = value('8K4')
          when bpool = 105 then
                bpool = value('8K5')
          when bpool = 106 then
                bpool = value('8K6')
          when bpool = 107 then
                bpool = value('8K7')
          when bpool = 108 then
                bpool = value('8K8')
          when bpool = 109 then
                bpool = value('8K9')
          otherwise
             nop
       end
    when bpool < 130 then
       select
          when bpool = 120 then
                bpool = value('16K0')
          when bpool = 121 then
                bpool = value('16K1')
          when bpool = 122 then
                bpool = value('16K2')
          when bpool = 123 then
                bpool = value('16K3')
          when bpool = 124 then
```

```
               bpool = value('16K4')
          when bpool = 125 then
               bpool = value('16K5')
          when bpool = 126 then
               bpool = value('16K6')
          when bpool = 127 then
               bpool = value('16K7')
          when bpool = 128 then
               bpool = value('16K8')
          when bpool = 129 then
               bpool = value('16K9')
          otherwise
             nop
       end
    otherwise
       nop
end

P1  = bpool
P2  = vpsize
P3  = hpsize
P4  = vpseqt
P5  = vppseqt
P6  = vpxpseqt
P7  = hpseqt
P8  = dwqt
P9  = vdwqt
P1Ø = castout
P11 = vptype
P12 = pgsteal

ADDRESS "ISPEXEC" "TBADD BUFLIST"
if rc ¬= Ø then do
   say 'error doing TBADD, RC=' rc
   exit rc
end
return
```

DB2BUFFP

```
)ATTR
/*--------------------------------------------------------------*/
/* DB2BUFFR - DISPLAY BUFFER INFO FROM BSDS KEY X'ØAØØØØØ1'       */
/*--------------------------------------------------------------*/
+ TYPE(TEXT)   INTENS(LOW)  COLOR(BLUE)   SKIP(ON)
% TYPE(TEXT)   INTENS(HIGH) COLOR(WHITE)  SKIP(ON)
# TYPE(OUTPUT) INTENS(HIGH) COLOR(YELLOW) CAPS(ON)
)BODY CMD(C)
%------------------+DISPLAY BUFFER POOL INFORMATION%-------------------+
%OPTION ===>_C                               %DB2 SSID ===>+#P0
+
%   B     VP      HP       VP     VP    VPX     HP             CAST    VP     PG
%   P    SIZE    SIZE     SEQT  PSEQT  PSEQT   SEQT  DWQT  VDWQT OUT   TYPE  STEAL
+------------------------------------------------------------------------+
)MODEL
_A+#P1 +#P2    +#P3    +#P4   +#P5   +#P6   +#P7   +#P8 +#P9 +#P10 +#P11 +#P12
+------------------------------------------------------------------------+
)INIT
)PROC
 VER (&C LIST,END,' ')
)END
```

DISPLAY

```
--------------- DISPLAY BUFFER POOL INFORMATION -------- Row 1 to 7 of 7
OPTION ===>                                      DB2 SSID ===>  DB2X

   B     VP      HP       VP     VP    VPX     HP             CAST    VP     PG
   P    SIZE    SIZE     SEQT  PSEQT  PSEQT   SEQT  DWQT  VDWQT OUT   TYPE  STEAL
-------------------------------------------------------------------------
   0    15600   0         80     50     0      80     50    10    Y      P      L
-------------------------------------------------------------------------
   1    17500   0         79     49     52     81     51    11    Y      P      L
-------------------------------------------------------------------------
   7     4369   8738      17     34     85     51     68    67    Y      P      L
-------------------------------------------------------------------------
   10    1000   0         80     50     0      80     50    10    Y      P      L
-------------------------------------------------------------------------
   32K   32     0         80     50     0      80     50    10    Y      P      L
-------------------------------------------------------------------------
   8K0   8      0         80     50     0      80     50    10    Y      P      L
-------------------------------------------------------------------------
   16K0  16     0         80     50     0      80     50    10    Y      P      L
-------------------------------------------------------------------------
```

(Note: VP and HP values are for sample purposes only and not a reflection of the true environment.)

John Mustric
Lead Systems Engineer
NiSource (USA)

The hierarchical structure of the DTD and XML document with mapping DTDs to DB2 – part 2

This month we conclude the code to map DTDs to DB2.

```
        } else {
          i = s1.indexOf("ATTLIST");
          if ((s1.indexOf("CDATA") > Ø) | (s1.indexOf('(') > Ø)) {
            if (s1.indexOf("CDATA") > Ø) {
              j = s1.indexOf("CDATA");
             } else {
              j = s1.indexOf('(');
            }
     k = addarr(n, k, "a." + s1.substring(i + 7, j).trim(), hlp3);
            if (s1.indexOf('>') < Ø) {
              for (kdtd = jdtd + 1 ; kdtd <= kdtd1; kdtd++) {
                     s1 = ndtd1[kdtd];
          if ((s1.indexOf("CDATA") > Ø) | (s1.indexOf('(') > Ø)) {
                  if (s1.indexOf("CDATA") > Ø) {
                    j = s1.indexOf("CDATA");
                   } else {
                    j = s1.indexOf('(');
                  }
         k = addarr(n, k, "a." + s1.substring(Ø, j).trim(), hlp3);
                }
                if (s1.indexOf('>') >= Ø) {
                  jdtd = kdtd;
                  break;
              }
            }
            }
           } else {      // definition like structure
            String hlp4 = new String();
            hlp4 = "e." + s1.substring(i + 7).trim();
            for (int l = Ø ; l <= k ; l++) {
                  if (n[l].indexOf(hlp4 + '+') > Ø) {
                hlp4 = hlp4 + '+';
              }
                  if (n[l].indexOf(hlp4 + '*') > Ø) {
                hlp4 = hlp4 + '*';
              }
            }
              String s2 = new String();
            for (kdtd = jdtd + 1 ; kdtd <= kdtd1; kdtd++) {
                   s2 = ndtd1[kdtd];
          if ((s2.indexOf("CDATA") > Ø) | (s2.indexOf('(') > Ø)) {
                if (s2.indexOf("CDATA") > Ø) {
```

```
            j = s2.indexOf("CDATA");
           } else {
            j = s2.indexOf('(');
          }
   k = addarr(n, k, "a." + s2.substring(0, j).trim(), hlp4);
        }
        if (s2.indexOf('>') >= 0) {
          jdtd = kdtd;
          break;
        }
      }  // end for
    }  // end if ... else
  }  // end if
}   // end if
}      // end for
// Exclude repeated elements
for (i = 0 ; i <= k; i++) {
  String rightp = n[i].substring(n[i].indexOf('-') + 1);
boolean fnd = false;
  for (j = 0 ; j <= k; j++) {
    String leftp = n[j].substring(0, n[j].indexOf('-'));
  if (rightp.equals(leftp)) {
    fnd = true;
    break;
    }
  }
if (!fnd) {
     try {
     in1 = new BufferedReader(new FileReader(fileSelected));
      s = new String();
    int nrpt = 0;
    int lineno = 0;
    nextwhile:
      while ((s = in1.readLine()) != null) {
        lineno++;
      if (s.indexOf("<!DOC") >= 0) {
      if (s.indexOf("[") >= 0) {
            while ((s = in1.readLine()) != null) {
            lineno++;
            if (s.indexOf("]") >= 0) break;
              }
        }
      continue nextwhile;
      }
 if (rightp.substring(0, 1).equals("a")) {      // attribute
      if (s.indexOf(rightp.substring(2)) >= 0) {
        nrpt++;
int ival = s.indexOf(rightp.substring(2)) + rightp.length();
        String hlpval = s.substring(ival);
              int jval = ival + hlpval.indexOf('"');
```

```
                hlpval = String.valueOf(nrpt) + ") " + rightp + "-v." +
s.substring(ival, jval);
                    il++;
                    nleaf[il] = String.valueOf(lineno) + ") " + hlpval;
                  } else {
if ((rightp.substring(2).indexOf('?') > Ø) & (s.indexOf("<!--") >  Ø)) {
                      nrpt++;
                            int ival = s.indexOf("<!--") + 4;
                            int jval = s.indexOf("-->");
                      String hlpval = String.valueOf(nrpt) + ") " +
rightp + "-v." + s.substring(ival, jval).trim();
                      il++;
                    nleaf[il] = String.valueOf(lineno) + ") " + hlpval;
                      }
                    }
                  } else {      // element
                  if (s.indexOf('<' + rightp.substring(2)) >= Ø) {
                    nrpt++;
     int ival = s.indexOf('<' + rightp.substring(2)) + rightp.length();
                          int jval = s.indexOf("</");
                    String hlpval = String.valueOf(nrpt) + ") " + rightp
+ "-v." + s.substring(ival, jval);
                    il++;
                    nleaf[il] = String.valueOf(lineno) + ") " + hlpval;
                  } else {
                    if ((rightp.substring(2).indexOf('?') > Ø) &
(s.indexOf("<!--") >  Ø)) {
                      nrpt++;
                            int ival = s.indexOf("<!--") + 4;
                            int jval = s.indexOf("-->");
                      String hlpval = String.valueOf(nrpt) + ") " +
rightp + "-v." + s.substring(ival, jval).trim();
                      il++;
                    nleaf[il] = String.valueOf(lineno) + ") " + hlpval;
                      }
                    }
                  }
                  }      // End while
                  in1.close();
                  } catch (FileNotFoundException e3) {
   mess = new MsgBox(new JFrame("") , "File Not Found:" + fileSelected);
                  } catch(IOException e3) {
                mess = new MsgBox(new JFrame("") , "IO Exception");
                  }
              }
            }
            // Add leaf elem/attr - value in array of edges
            int kk = k;
            for (i = Ø ; i <= il; i++) {
            boolean fnd = false;
```

```
             String lpar = nleaf[i].substring(nleaf[i].lastIndexOf(')') +
2, nleaf[i].indexOf('-'));
               for (j = Ø ; j <= k; j++) {
              String nchild = n[j].substring(n[j].indexOf('-') + 1);
              String npar = n[j].substring(Ø, n[j].indexOf('-'));
              if (nchild.equals(lpar)) {
                if ((npar.indexOf('+') > Ø) | (npar.indexOf('*') > Ø)) {
                  fnd = true;
                  break;
                  }
                }
              }
            if (!fnd) {
              kk++;
              n[kk] = nleaf[i].substring(nleaf[i].lastIndexOf(')') + 2);
              nleaf[i] = "***" + nleaf[i];
              }
            }
            String nleaf1[] = new String[5ØØ];
            k = -1;
            for (i = Ø; i <= il; i++) {
               for (j = Ø; j < i; j++) {
              if (nleaf[i].equals(nleaf[j])) {
                nleaf[i] = "***" + nleaf[i];
                break;
                }
              }
            if (!nleaf[i].substring(Ø, 3).equals("***")) {
              k++;
              nleaf1[k] = nleaf[i];
              }
            }
             // prepare combo boxes
            il = k;
            int lvl = 1;
            icb = -1;
            for (i = Ø; i <= kk; i++) {
            String nchild = n[i].substring(n[i].indexOf('-') + 3);
            String npar = n[i].substring(2, n[i].indexOf('-'));
            if ((nchild.indexOf('+') > Ø) | (nchild.indexOf('*') > Ø)) {
              if ((npar.indexOf('+') > Ø) | (npar.indexOf('*') > Ø)) {
                      for (j = Ø; j <= icb; j++) {
         if (ncb[j].indexOf(npar.substring(Ø, npar.length() - 1)) > Ø) {
   lvl = Integer.parseInt(ncb[j].substring(Ø, ncb[j].indexOf(')'))) + 1;
                    icb++;
                    ncb[icb] = String.valueOf(lvl) + ')' +
prepcombo(fileSelected, nchild.substring(Ø, nchild.length() - 1));
                  break;
                    }
                  }
                 } else {
                lvl = 1;
                icb++;
                ncb[icb] = String.valueOf(lvl) + ')' +
prepcombo(fileSelected, nchild.substring(Ø, nchild.length() - 1));
              }
              }
            }
            for (i = Ø; i <= icb; i++) {
              for (j = Ø; j < 5ØØ; j++) {
              matrix[i][j] = " ";            // CB values
              }
            }
            maxp = -1;
            int maxpos[][] = new int[icb + 1][5ØØ];
            for (i = Ø; i <= icb; i++) {
            if (ncb[i].indexOf(':') < Ø) {
              ncb[i] = ncb[i] + ":Ø-1ØØØ";
                }
            if (i == Ø) {
    minncb = Integer.parseInt(ncb[i].substring(ncb[i].indexOf(':') + 1,
ncb[i].indexOf('-')));
                }
            String cbname = ncb[i].substring(ncb[i].indexOf(')') + 1,
ncb[i].indexOf(':'));
              for (j = Ø; j <= kk; j++) {
          if (n[j].substring(2, n[j].indexOf('-') - 1).equals(cbname)) {
                 String rname = n[j].substring(n[j].indexOf('-') + 3);
                      for (k = Ø; k <= il; k++) {
                    if (nleaf1[k].substring(nleaf1[k].lastIndexOf(')') +
4, nleaf1[k].indexOf('-')).equals(rname)) {
                    int pos = Integer.parseInt(nleaf1[k].substring(Ø,
nleaf1[k].indexOf(')')));
       int p = findpos(pos, ncb[i].substring(ncb[i].indexOf(':') + 1));
                  if (p > maxp) {
                    maxp = p;
                      }
                  maxpos[i][p] = pos;
                  if (matrix[i][p].indexOf(rname) < Ø ) {
                     matrix[i][p] = matrix[i][p] + rname + ':' +
nleaf1[k].substring(nleaf1[k].indexOf('-') + 3) + ';';
                  } else {
                     matrix[i][p] = matrix[i][p].substring(Ø,
matrix[i][p].lastIndexOf(':') + 1);
                     matrix[i][p] = matrix[i][p] +
nleaf1[k].substring(nleaf1[k].indexOf('-') + 3) + ';';
                  }
                  // update previous records
                  for (int ip = Ø; ip <= maxp; ip++) {
                    if (matrix[i][ip].indexOf(rname) < Ø ) {
```

```
                      matrix[i][ip] = matrix[i][ip] + rname + ": ;";
                        }
                      }
                    }
                  } // for k
                }
              } // for j
            } // for i
            // Insert new items (leaf) from non-repeated lists (items
with same name, and position in Xml < min ncb)
            for (i = Ø; i <= il; i++) {
            if (Integer.parseInt(nleaf1[i].substring(Ø,
nleaf1[i].indexOf(')'))) < minncb) {
      String renobjfrom = nleaf1[i].substring(nleaf1[i].lastIndexOf(')')
+ 2, nleaf1[i].indexOf('-'));
              String renobjto = "";
                for (j = Ø; j <= il; j++) {
    if (n[j].substring(n[j].lastIndexOf('-') + 1).equals(renobjfrom)) {
                  renobjto = "renxxren" + String.valueOf(j);
                  n[j] = n[j] + renobjto;
                  break;
                }
              }
              kk++;
              n[kk] = nleaf1[i].substring(nleaf1[i].lastIndexOf(')') +
2, nleaf1[i].indexOf('-')) + renobjto +
nleaf1[i].substring(nleaf1[i].indexOf('-'));
            }
            }
            // add key in combo boxes on the end of items
            for (i = Ø; i <= icb; i++) {
              for (j = Ø; j <= maxp; j++) {
              if (!matrix[i][j].equals(" ")) {
                int pcb = Ø;
                if (i > Ø) {
                  pcb = findpos(maxpos[i][j], ncb[i-1].substring(ncb[i-
1].indexOf(':') + 1));
                  } else {
                  pcb = findpos(maxpos[i][j],
ncb[Ø].substring(ncb[Ø].indexOf(':') + 1));
                  }
                matrix[i][j] = matrix[i][j] + String.valueOf(pcb);
                // Insert new items (leaf) from repeated lists
                if (j == Ø) {
                  String elatt = matrix[i][j].substring(Ø,
matrix[i][j].lastIndexOf(';')).trim();
                  for (StringTokenizer t = new StringTokenizer(elatt,
";") ; t.hasMoreTokens() ; ) {
                      String elattstr = t.nextToken();
           String lelatt = elattstr.substring(Ø, elattstr.indexOf(':'));
```

```
                  String str1 = "e.";
                  for (int iea = Ø; iea <= kk; iea++) {
          if (n[iea].substring(n[iea].indexOf('-') + 3).equals(lelatt) &
                        n[iea].substring(2, n[iea].indexOf('-') -
1).equals(ncb[i].substring(ncb[i].indexOf(')') + 1,
ncb[i].indexOf(':')))) {
                       str1 = n[iea].substring(n[iea].indexOf('-') + 1,
n[iea].indexOf('-') + 3);
                      break;
                    }
                  }
                  lelatt = str1 + lelatt;
  String relatt = "v." + elattstr.substring(elattstr.indexOf(':') + 1);
                  kk++;
                  n[kk] = lelatt + '-' + relatt;
                    }
                  }
                }
              }
            }
            // remove ? from name
            for (i = Ø; i <= icb; i++) {
              for (j = Ø; j <= maxp; j++) {
                while (matrix[i][j].indexOf('?') > Ø) {
                int m = matrix[i][j].indexOf('?');
                matrix[i][j] = matrix[i][j].substring(Ø, m) +
matrix[i][j].substring(m + 1);
                }
              }
            }
            for (i = Ø; i <= kk; i++) {
              while (n[i].indexOf('?') > Ø) {
              int m = n[i].indexOf('?');
              n[i] = n[i].substring(Ø, m) + n[i].substring(m + 1);
              }
            }
            // tree sort
            String Root = new String();
            for (i = Ø; i <= kk; i++) {
            boolean fnd = false;
            String Left = n[i].substring(Ø, n[i].indexOf('-')).trim();
              for (j = Ø; j <= kk; j++) {
        if (n[j].substring(n[j].indexOf('-') + 1).trim().equals(Left)) {
                fnd = true;
                break;
                }
              }
            if (!fnd) {
              Root = Left;
              break;
              }
            }
            kk1 = -1;
            kk2 = 1;
            String n1[] = new String[300];
            sorttree(1, Root, n, kk, n1);
            int n3[] = new int[kk2 + 1];
            for (i = 0; i <= kk2; i++) {
            n3[i] = 0;
            }
            for (i = 0; i < kk1; i++) {
            String child = n1[i].substring(n1[i].indexOf('-') + 1);
            int rowstart = Integer.parseInt(n1[i].substring(0,
n1[i].indexOf(')')));
              for (j = (i + 1); j <= kk1; j++) {
              String parent = n1[j].substring(n1[j].indexOf(')') + 1,
n1[j].indexOf('-'));
              int row = Integer.parseInt(n1[j].substring(0,
n1[j].indexOf(')')));
              if (parent.equals(child) & row == rowstart + 1) {
                n3[row]++;
                n1[j] = n1[j].substring(0, n1[j].indexOf(')') + 1) +
String.valueOf(n3[row]) + ')' + n1[j].substring(n1[j].indexOf(')') + 1);
                }
              }
            }
            j = 0;
            for (i = 0; i <= kk1; i++) {
                if (n1[i].substring(n1[i].indexOf(')') + 1,
n1[i].indexOf('-')).equals(Root)) {
                j++;
                n1[i] = n1[i].substring(0, n1[i].indexOf(')') + 1) +
String.valueOf(j) + ')' + n1[i].substring(n1[i].indexOf(')') + 1);
                }
            }
            // coordinates
            int maxcolroot = 2;
            for (i = 0; i < kk1; i++) {
            String child = n1[i].substring(n1[i].indexOf('-') + 1);
            int colchild =
Integer.parseInt(n1[i].substring(n1[i].indexOf(')') + 1,
n1[i].lastIndexOf(')')));
            int rowchild = Integer.parseInt(n1[i].substring(0,
n1[i].indexOf(')')));
            int mincol = 999;
            int maxcol = -1;
            boolean fnd = false;
              for (j = (i + 1); j <= kk1; j++) {
              String parent = n1[j].substring(n1[j].lastIndexOf(')') +
1, n1[j].indexOf('-'));
```

```
              int colpar =
Integer.parseInt(n1[j].substring(n1[j].indexOf(')') + 1,
n1[j].lastIndexOf(')')));
  int rowpar = Integer.parseInt(n1[j].substring(Ø, n1[j].indexOf(')')));
              if (parent.equals(child)) {
                fnd = true;
                if ((j == (i + 1)) & (colpar < colchild)) {
                  int diff = colchild - colpar;
                    for (k = j; k <= kk1; k++) {
 int rowpar1 = Integer.parseInt(n1[k].substring(Ø, n1[k].indexOf(')')));
                    int colpar1 =
Integer.parseInt(n1[k].substring(n1[k].indexOf(')') + 1,
n1[k].lastIndexOf(')')));
                    if (rowpar == rowpar1) {
                    colpar1 = colpar1 + diff;
                        n1[k] = n1[k].substring(Ø, n1[k].indexOf(')') +
1) + String.valueOf(colpar1) + ')' +
n1[k].substring(n1[k].lastIndexOf(')') + 1);
                      }
                    }
                  colpar = colchild;
                  }
                if (colpar < mincol) {
                  mincol = colpar;
                  }
                if (colpar > maxcol) {
                  maxcol = colpar;
                  }
                }
              }
            if (fnd) {
              int avgcol = (mincol + maxcol) / 2;
              if (avgcol > colchild) {
                int diff = avgcol - colchild;
                  for (k = i; k <= kk1; k++) {
                  int rowchild1 = Integer.parseInt(n1[k].substring(Ø,
n1[k].indexOf(')')));
                  int colchild1 =
Integer.parseInt(n1[k].substring(n1[k].indexOf(')') + 1,
n1[k].lastIndexOf(')')));
                  if (rowchild == rowchild1) {
                  colchild1 = colchild1 + diff;
                      n1[k] = n1[k].substring(Ø, n1[k].indexOf(')') + 1)
+ String.valueOf(colchild1) + ')' +
n1[k].substring(n1[k].lastIndexOf(')') + 1);
                  if (colchild1 > maxcolroot) {
                    maxcolroot = colchild1;
                    }
                    }
                  }
                }
              }
            }
            // create graph
            for (i = 0; i <= kk1; i++) {
     int row = Integer.parseInt(n1[i].substring(0, n1[i].indexOf(')')));
      int col = Integer.parseInt(n1[i].substring(n1[i].indexOf(')') + 1,
n1[i].lastIndexOf(')')));
            String parent = n1[i].substring(n1[i].lastIndexOf(')') + 1,
n1[i].indexOf('-'));
            String rightparm = addparm(row + 1, col);
            String leftparm;
            if (parent.equals(Root)) {
              leftparm = addparm(row, maxcolroot / 2);
              } else {
              leftparm = findparm(parent, n1, i, row);
              }
            n1[i] = n1[i] + "|" + leftparm + rightparm;
            }
            for (i = 0; i <= kk1; i++) {
            n1[i] = n1[i].substring(n1[i].lastIndexOf(')') + 1);
            }
            XmlDtdPanel.nnodes = 0;
            XmlDtdPanel.nedges = 0;
            for (i = 0; i <= kk1; i++) {
              String str = n1[i];
              int ii = str.indexOf('-');
              if (ii > 0) {
              j = str.indexOf('|');
              String posx = str.substring(j+1,j+5);
              String posy = str.substring(j+5);
              str = str.substring(0, j);
              String typefrom = str.substring(0,1);
              if (str.substring(0,ii).indexOf('*') > 0 |
str.substring(0,ii).indexOf('+') > 0) {
                typefrom = typefrom + '*';
              }
              String typeto = str.substring(ii+1,ii+2);
              if (str.substring(ii+1).indexOf('*') > 0 |
str.substring(ii+1).indexOf('+') > 0) {
                typeto = typeto + '*';
              }
              while (str.indexOf('*') > 0 | str.indexOf('+') > 0) {
                if (str.indexOf('*') > 0) {
                  k = str.indexOf('*');
                } else {
                  k = str.indexOf('+');
                }
                if (!(str.indexOf("v.") > 0 & k > str.indexOf("v."))) {
                  str = str.substring(0, k) + str.substring(k + 1);
                  ii = str.indexOf('-');
                }
              }
              panel.addEdge(typefrom, str.substring(2,ii), typeto,
str.substring(ii+3), 5Ø, posx, posy);
              }
            }
          } catch (FileNotFoundException e1) {
   mess = new MsgBox(new JFrame("") , "File Not Found:" + fileSelected);
          } catch (IOException e1) {
              mess = new MsgBox(new JFrame("") , "IO Exception");
          }
          }
        return;
      } else {
        if (src == mapping_button) {
          int i, j, k, l;
          String pkey, pkey1, mstr;
           try {
            BufferedWriter out = new BufferedWriter(new
FileWriter("creatab.sql"));
              String newline = System.getProperty("line.separator");
            pkey1 = "";
              for (i = Ø; i < XmlDtdPanel.nnodes; i++) {
              if (XmlDtdPanel.nodes[i].type.indexOf('*') > Ø |
XmlDtdPanel.nodes[i].lbl.equals(XmlDtdPanel.nodes[i].parent)) {
              if (XmlDtdPanel.nodes[i].lbl.indexOf(' ') > Ø)
out.write("CREATE TABLE \"" + XmlDtdPanel.nodes[i].lbl + "\" (" +
newline);
              else out.write("CREATE TABLE " + XmlDtdPanel.nodes[i].lbl
+ " (" + newline);
                mstr = "  ";
              pkey = "";
                  for (j = i + 1; j < XmlDtdPanel.nnodes; j++) {
                  if
(XmlDtdPanel.nodes[j].parent.indexOf(XmlDtdPanel.nodes[i].lbl) >= Ø &
              !(XmlDtdPanel.nodes[j].type.substring(Ø, 1).equals("v"))) {
            if (XmlDtdPanel.nodes[j].type.substring(Ø, 1).equals("a")) {
                        out.write("   " + mstr +
finddatatype(XmlDtdPanel.nodes[j].lbl) + newline);
                      mstr = ", ";
                    pkey = pkey + XmlDtdPanel.nodes[j].lbl + ";";
                      } else {
                        if (XmlDtdPanel.nodes[j].type.indexOf('*') < Ø &
XmlDtdPanel.nodes[i].lbl.equals(XmlDtdPanel.nodes[i].parent)) {
                          l = Ø;
                          for (k = j + 1; k < XmlDtdPanel.nnodes; k++) {
                          if
(XmlDtdPanel.nodes[k].parent.indexOf(XmlDtdPanel.nodes[j].lbl) >= Ø &
                          XmlDtdPanel.nodes[k].type.indexOf('*') < Ø &
            !(XmlDtdPanel.nodes[k].type.substring(Ø, 1).equals("v"))) {
                       l++;
                          out.write("   " + mstr +
finddatatype(XmlDtdPanel.nodes[k].lbl) + newline);
                         mstr = ", ";
                         }
                        }
                    if (l == Ø) {
                         out.write("   " + mstr +
finddatatype(XmlDtdPanel.nodes[j].lbl) + newline);
                          mstr = ", ";
                        }
                       } else {
                      if (XmlDtdPanel.nodes[j].type.indexOf('*') < Ø) {
                         out.write("   " + mstr +
finddatatype(XmlDtdPanel.nodes[j].lbl) + newline);
                        mstr = ", ";
                        }
                       }
                      }
                     }
                   }
             if (pkey1 != "") {
                  for (StringTokenizer t = new StringTokenizer(pkey1,
";") ; t.hasMoreTokens() ; ) {
                      String t1 = t.nextToken();
                  out.write("   " + mstr + finddatatype(t1) + newline);
                    mstr = ", ";
                  }
                  }
                  out.write(");" + newline);
             pkey1 = pkey;
               }
             }
           out.close();
            mess = new MsgBox(new JFrame("") , "CRETAB.SQL file is
created in current directory!!!");
          } catch (IOException ew) {
             mess = new MsgBox(new JFrame("") , "IO Exception");
          }
          }
        }
    }
    String finddatatype(String str) {
      String str1 = "";
      String str2 = "";
      int j;
      for (int i = Ø; i < XmlDtdPanel.nnodes; i++) {
      if (XmlDtdPanel.nodes[i].parent.equals(str)) {
          j = Ø;
        if (isNumeric(XmlDtdPanel.nodes[i].lbl)) {
    if (XmlDtdPanel.nodes[i].lbl.indexOf('e') > 0) str1 = " float(16)";
          else if (XmlDtdPanel.nodes[i].lbl.indexOf('-') > 0 &
XmlDtdPanel.nodes[i].lbl.indexOf('.') > 0) str1 = " timestamp";
               else if (XmlDtdPanel.nodes[i].lbl.indexOf('-') > 0 |
XmlDtdPanel.nodes[i].lbl.indexOf('/') > 0 |
                         (XmlDtdPanel.nodes[i].lbl.lastIndexOf('.') >
XmlDtdPanel.nodes[i].lbl.indexOf('.'))) str1 = " date";
              else if (XmlDtdPanel.nodes[i].lbl.indexOf('.') > 0) j = 1;
     else if (XmlDtdPanel.nodes[i].lbl.indexOf(':') > 0) str1 = " time";
                              else j = 2;
          } else j = 3;
        if (j > 0) {
          str2 = XmlDtdPanel.nodes[i].lbl;
            for (int i1 = 0; i1 < icb + 1; i1++) {
             for (int j1 = 0; j1 < maxp + 1; j1++) {
                int k1 = matrix[i1][j1].indexOf(str);
            if (k1 >= 0) {
              int k2 = matrix[i1][j1].indexOf(':', k1);
              if (matrix[i1][j1].substring(k2 + 1,
matrix[i1][j1].indexOf(';', k2)).length() >
XmlDtdPanel.nodes[i].lbl.length()) {
                str2 = matrix[i1][j1].substring(k2 + 1,
matrix[i1][j1].indexOf(';', k2));
                  }
                }
              }
            }
          }
          switch(j) {
          case 1:
          if (str2.length() > 15) str1 = " char(" + str2.length() + ")";
            else str1 = " dec(" + (str2.length() - 1) + "," +
str2.substring(str2.indexOf('.') + 1).length() + ")";
            break;
          case 2:
         if (str2.length() > 15) str1 = " char(" + str2.length() + ")";
       else if (str2.length() > 9) str1 = " dec(" + str2.length() + ")";
                 else if (Integer.parseInt(str2) > 32768) str1 = " int";
                      else str1 = " smallint";
            break;
          case 3:
       if (str2.length() > 50) str1 = " varchar(" + str2.length() + ")";
      else if (str2.length() > 0) str1 = " char(" + str2.length() + ")";
                 else str1 = " char(1)";
            break;
          default:
            break;
          }
        if (str.indexOf(' ') > 0) str = "\"" + str + "\"" + str1;
```

```
          else str = str + str1;
        }
      }
      return str;
    }
    boolean isNumeric (String s) {
      boolean fnd = true;
      if (s.length() <= Ø) fnd = false;
      for (int i = Ø; i < s.length(); i++) {
        if (!((s.charAt(i) >= 'Ø' & s.charAt(i) <='9') | s.charAt(i) ==
'.' | s.charAt(i) == ':' | s.charAt(i) == '/' | s.charAt(i) == '-' |
s.charAt(i) == 'e')) {
          fnd = false;
        break;
        }
      }
      if (fnd) return true;
      else return false;
    }
    public void sortarray(int lvl, String rootdtd, int inddtd, String[]
ndtd, String[] ndtd1) {
      boolean fnd = false;
      int indpar = 9999;
      int retval = -1;
      for (int k = Ø ; k <= inddtd; k++) {
        if (!ndtd[k].substring(Ø, 1).equals("*")) {
        if (ndtd[k].indexOf('(') > Ø) indpar = ndtd[k].indexOf('(');
 if (ndtd[k].indexOf(rootdtd) > Ø & ndtd[k].indexOf(rootdtd) < indpar) {
 if ((ndtd[k].indexOf("<!ELEMENT") >= Ø) & (ndtd[k].indexOf('(') >= Ø) &
           (ndtd[k].indexOf(')') >= Ø) & (ndtd[k].indexOf('#') < Ø)) {
            fnd = true;
            retval = k;
          }
          kdtd1++;
          ndtd1[kdtd1] = ndtd[k];
          ndtd[k] = "*" + ndtd[k];
          if (ndtd[k].indexOf('>') < Ø) {
            for (int l = k + 1 ; l <= inddtd; l++) {
              kdtd1++;
              ndtd1[kdtd1] = ndtd[l];
            ndtd[l] = "*" + ndtd[l];
              if (ndtd[l].indexOf('>') >= Ø) {
              break;
              }
            }
          }
        }
      }
      }
      if (fnd) {
        for (StringTokenizer t = new
StringTokenizer(ndtd[retval].substring(ndtd[retval].indexOf('(') + 1,
ndtd[retval].indexOf(')')), ",") ; t.hasMoreTokens() ; ) {
          String t1 = t.nextToken();
        if (t1.indexOf('*') > Ø) t1 = t1.substring(Ø, t1.indexOf('*'));
        if (t1.indexOf('+') > Ø) t1 = t1.substring(Ø, t1.indexOf('+'));
        if (t1.indexOf('?') > Ø) t1 = t1.substring(Ø, t1.indexOf('?'));
          sortarray(lvl + 1, t1, inddtd, ndtd, ndtd1);
        }
      }
      return;
    }
    public String addparm(int r, int c) {
      String ret;
      if (r <= 9) {
        ret = "Ø" + String.valueOf(r);
      } else {
        ret = String.valueOf(r);
      }
      if (c <= 9) {
        ret = ret + "Ø" + String.valueOf(c);
      } else {
        ret = ret + String.valueOf(c);
      }
      return ret;
    }
 public String findparm(String child, String[] n, int j, int rowchild) {
      String ret = "ØØØØ";
      for (int i = Ø; i <= j; i++) {
        int rowparent = Integer.parseInt(n[i].substring(Ø,
n[i].indexOf(')')));
        if (n[i].substring(n[i].indexOf('-') + 1,
n[i].indexOf('|')).equals(child) &
            rowparent == rowchild - 1) {
          return n[i].substring(n[i].indexOf('|') + 5);
        }
      }
      return ret;
    }
    public void sorttree(int lvl1, String str, String[] n, int kk,
String[] n1) {
      int i;
      if (lvl1 > kk2) {
        kk2 = lvl1;
      }
      for (i = Ø; i <= kk; i++) {
        if (n[i].substring(Ø, n[i].indexOf('-')).equals(str)) {
          kk1++;
          n1[kk1] = lvl1 + ")" + n[i];
   sorttree(lvl1 + 1, n[i].substring(n[i].indexOf('-') + 1), n, kk, n1);
```

```
        }
      }
      return;
    }
    public int findpos(int pos, String str1) {
      int j = -1;
      for (StringTokenizer t = new StringTokenizer(str1, ":") ;
t.hasMoreTokens() ; ) {
          j++;
          String str = t.nextToken();
          int i = str.indexOf('-');
          int frpos = Integer.parseInt(str.substring(Ø, i));
          int topos = Integer.parseInt(str.substring(i + 1));
          if ((pos >= frpos) & (pos <= topos)) {
            break;
          }
      }
      return j;
    }
    public String prepcombo(String fileSel, String str) {
      String s = new String();
      String scb = str + ':';
      int lineno = Ø;
      int lfrom = Ø;
      int lto = Ø;
      try {
       BufferedReader inc = new BufferedReader(new FileReader(fileSel));
      while ((s = inc.readLine()) != null) {
        lineno++;
        if (s.indexOf('<' + str) >= Ø) {
          lfrom = lineno;
        }
        if (s.indexOf("</" + str) >= Ø) {
          lto = lineno;
    scb = scb + String.valueOf(lfrom) + '-' + String.valueOf(lto) + ':';
        }
          }
      inc.close();
      scb = scb.substring(Ø, scb.length() - 1);
      return scb;
      } catch (FileNotFoundException e) {
       mess = new MsgBox(new JFrame("") , "File Not Found:" + fileSel);
      return scb;
      } catch(IOException e) {
       mess = new MsgBox(new JFrame("") , "IO Exception");
      return scb;
      }
    }
    int addarr(String s[], int i, String s1, String s2) {
      for (int j = Ø ; j < i ; j++) {
          if (s[j].indexOf(s2 + "-" + s1) > 0) {
            return i;
          }
      }
      i++;
      s[i] = s2 + '-' + s1;
      return i;
    }
    public String getAppletInfo() {
      return "Title: XmlDtdLayout \nAuthor: Nikola Lazovic \n\t Postal
Savings Bank\n\t Serbia and Montenegro";
    }
    public String[][] getParameterInfo() {
      String[][] info = {
          {"edges", "delimited string", "A comma-delimited list of all
the edges.\n It takes the form of 'PP1-CN1|RPCPRCCC,PP2-
CN2|RPCPRCCC,...PPX-CNX|RPCPRCCC'\n\t where PPX is the name of parent
node, \n\t CNX is the name of child node, \n\t RPCP is row number and
column number of parent node and \n\t RCCC is row number and column
number of child node."}
      };
      return info;
    }
}
```

MYFILEFILTER.JAVA

```
/* ******************************************************* */
/* MyFileFilter.java                                        */
/*    Open file dialog                                      */
/* ******************************************************* */
import java.io.File;
import java.util.Hashtable;
import java.util.Enumeration;
import javax.swing.*;
import javax.swing.filechooser.*;
public class MyFileFilter extends FileFilter {
    private static String TYPE_UNKNOWN = "Type Unknown";
    private static String HIDDEN_FILE = "Hidden File";
    private Hashtable filters = null;
    private String description = null;
    private String fullDescription = null;
    private boolean useExtensionsInDescription = true;
    public MyFileFilter() {
      this.filters = new Hashtable();
    }
    public MyFileFilter(String extension) {
      this(extension,null);
    }
    public MyFileFilter(String extension, String description) {
      this();
      if(extension!=null) addExtension(extension);
       if(description!=null) setDescription(description);
    }
    public MyFileFilter(String[] filters) {
      this(filters, null);
    }
    public MyFileFilter(String[] filters, String description) {
      this();
      for (int i = 0; i < filters.length; i++) {
          addExtension(filters[i]);
      }
       if(description!=null) setDescription(description);
    }
    public boolean accept(File f) {
      if(f != null) {
          if(f.isDirectory()) {
            return true;
          }
          String extension = getExtension(f);
          if(extension != null && filters.get(getExtension(f)) != null)
{
            return true;
          };
      }
      return false;
    }
     public String getExtension(File f) {
      if(f != null) {
          String filename = f.getName();
          int i = filename.lastIndexOf('.');
          if(i>0 && i<filename.length()-1) {
            return filename.substring(i+1).toLowerCase();
          };
      }
      return null;
    }
    public void addExtension(String extension) {
      if(filters == null) {
          filters = new Hashtable(5);
      }
      filters.put(extension.toLowerCase(), this);
      fullDescription = null;
    }
    public String getDescription() {
      if(fullDescription == null) {
          if(description == null || isExtensionListInDescription()) {
         fullDescription = description==null ? "(" : description + " (";
            Enumeration extensions = filters.keys();
            if(extensions != null) {
            fullDescription += "." + (String) extensions.nextElement();
                while (extensions.hasMoreElements()) {
          fullDescription += ", ." + (String) extensions.nextElement();
                }
            }
            fullDescription += ")";
          } else {
            fullDescription = description;
          }
      }
      return fullDescription;
    }
    public void setDescription(String description) {
      this.description = description;
      fullDescription = null;
    }
    public void setExtensionListInDescription(boolean b) {
      useExtensionsInDescription = b;
      fullDescription = null;
    }
    public boolean isExtensionListInDescription() {
      return useExtensionsInDescription;
    }
}
```

XMLDTD.HTML

```
<html>
        <head>
      <title>The hierarchical structure of the DTD and XML document with
Mapping DTDs to DB2</title>
  </head>
  <body>
    <h3>The Hierarchical Structure of the DTD and XML Document with
Mapping DTDs to DB2</h3>
      <hr>
      <applet codebase="." code="XmlDtd.class" width=980 height=600>
      <param name=edges value="root-attr1*|01050201,root-
elem1|01050205,root-elem2|01050209,attr1*-val1|02010301,elem1-
attr2|02050302,elem1-elem5|02050303,elem1-elem6|02050304,elem1-
elem7|02050305,elem1-elem8|02050306,elem1-elem9*|02050307,elem2-
elem3|02090308,elem2-elem4|02090309,attr2-val4|03020402,elem5-
val5|03030403,elem6-val6|03040404,elem7-val7|03050405,elem8-
val8|03060406,elem9*-elem10|03070407,elem9*-elem11|03070408,elem3-
val2|03080409,elem4-val3|03090410,elem10-val9|04060506,elem11-
val10|04070507">
      alt="Your browser understands the &lt;APPLET&gt; tag but isn't
running the applet, for some reason."
```

```
        Your browser is completely ignoring the &lt;APPLET&gt; tag!
        </applet>
        <hr>
    </body>
</html>
```

Nikola Lazovic
DB2 System Administrator
Postal Savings Bank (Serbia and Montenegro)

DB2 UDB for LUW 8.2 – how to detect locking using the Activity Monitor

A common complaint made to DBAs is, “My application seems to be hanging and I don’t know why – help!”. So what should the DBA do? Well, with Version 8.2 (8.1.7) of DB2 UDB for LUW we now have the Activity Monitor in our armoury. Before this release, we would have had either to use a tool (such as Performance Expert or similar) or to run an event trace and analyse it (which was never easy at the best of times). The Activity Monitor gives us a graphical representation of any locking conflicts, which is extremely user friendly.

This article goes through the steps required to set up the Activity Monitor to find out what applications/userids are involved in a locking/deadlock situation. The SQL was run on a Windows 2000 Professional box running DB2 UDB 8.2.1 for LUW. I will use the SAMPLE database with the LOCKTIMEOUT database configuration parameter set to -1.

What we will do is:

- Set up the Activity Monitor to monitor locking.
- Create two userids (db2user1 and db2user2).
- Grant UPDATE authority on table db2admin.employee to db2user1 and db2user2.

- Update table db2admin.employee using db2user1, but not commit the changes.
- Update table db2admin.employee using db2user2.
- Go into the Activity Monitor to see whether we can see the locking conflict.

The first thing we need to do is configure the Activity Monitor. From the command line type **>db2am**. This takes you to the *Activity Monitor introduction screen*. You need to specify the database you want to monitor. In our case that is the SAMPLE database. You are then taken to the *Monitoring* screen. There are four system-defined activity monitors:

- Resolving a general database system slowdown
- Resolving the performance degradation of an application
- Resolving an application lock situation
- Tuning the dynamic SQL statement cache.

We will take a copy of the *Resolving an application lock situation* and call it 'Locking' – we do this by highlighting the *Resolving an application lock situation* line, clicking on **New** and giving it a name of 'Locking'. Clicking on **Next** takes us to the *Specify the category of monitoring task* screen – we just want to specify *Trouble shooting/Identify one or more problem applications*, **Next**, **All Applications**, tick all the 'Lock'-related boxes, and make sure the **Show application lock chains** box is ticked. Click **Next**, **Next**, and **Finish** and the Activity Monitor is set up!

Create the db2user1 and db2user2 userids (from the Windows control panel).

From db2admin issue:

```
>db2 grant update on table db2admin.employee to db2user1

>db2 grant update on table db2admin.employee to db2user2
```

Now issue the following commands in the sequence shown.

From db2user1 issue:

```
>db2 connect to sample user db2user1 using db2user1
>db2 +c "update db2admin.employee set salary = salary + 1"
```

The command will complete successfully and a prompt is returned.

From db2user2 issue:

```
>db2 connect to sample user db2user2 using db2user2
>db2 +c "update db2admin.employee set salary = salary + 1"
```

The command will hang.

From the db2admin userid, open the Activity Monitor from the Control Center bottom right-hand pane. Click on **Next**, highlight *Locking* under the **User-defined** tab, click **Finish**. For the Report, select **Applications holding the largest number of locks**.

Highlight the line that has an authorization ID of DB2USER2 (because this is the one that is hanging), right-click, and select **Show lock chains**. See Figure 1.

Press the **Legend** button to see what the boxes/lines mean; but basically the double lined box means this is where we issued the show command from, and the pointing arrow means that the double lined box is waiting for the box that the arrow is pointing to (DB2USER1 in our case).

Here are a couple of questions and answers.

Can you set up the Activity Monitor once locking is occurring? Yes you can – but I would set it up in advance, so that if you have a problem, you can quickly go to the **show lock chains** screen.

Does running the Activity Monitor cost? Yes – there is no such thing as a free lunch! There is a warning at the bottom of the Activity Monitor set-up screen saying that monitor switches are used to collect the appropriate data.

You can use the Activity Monitor in conjunction with the Health Center to monitor locking and discover/resolve problems

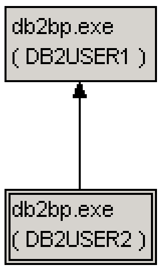

Figure 1: Display hanging task

before users start complaining too much! It is indeed a very useful addition to the DB2 family of tools and is a great answer to one of the most common questions/problems.

C Leonard
Freelance Consultant (UK)

DB2 news

Compuware has announced Version 3.2 of Strobe and Version 2.1 of iStrobe.

Strobe 3.2 features the new enhancement of DB2 DDF support, which helps users manage the performance of their distributed applications that access mainframe DB2.

iStrobe is a Web server application that analyses Strobe application performance data, allowing users to measure the performance of applications and identify problems.

For further information contact:
URL: www.compuware.com/products/strobe/default.htm.

* * *

Quest Software has announced Version 4.5 of Quest Central for DB2, its integrated database management product designed to increase the productivity of DB2 DBAs.

Through a single console, Quest Central for DB2 allows DBAs to perform database administration, SQL tuning, space management, and performance diagnostics on DB2 databases running on Linux, Unix, Windows, and z/OS.

For further information contact:
URL: www.quest.com/quest_central_for_db2.

* * *

nCipher Corp, which specializes in database encryption, has announced that its SecureDB product now supports DB2 and Microsoft SQL Server – previously it supported only Oracle databases.

SecureDB enables users to encrypt just the sensitive information in a database leaving non-sensitive information unencrypted. It includes a policy enforcement application and database analysis tool designed to streamline deployment and to selectively apply this extra layer of security in the most efficient manner.

SecureDB provides for a separation of duties designed to eliminate the 'super-user' threat by dividing authority between security and access.

For further information contact:
URL: www.livetime.com/WebHelpDesk/app?service=page/Press080305.

* * *

IBM has announced Version 8.3 of DB2 CommonStore for SAP, which helps users perform periodic archiving of old or infrequently accessed data. The software incorporates Linux platform support, annotation capability, and security features. The software allows users to reduce back-up/restore window cycle times, integrate digitized content in SAP applications, and establish a scalable and secure back-end archive management repository.

For further information contact your local IBM representative:
URL: www-306.ibm.com/software/data/commonstore/sap/

* * *

xephon