151

DB2

update

May 2005

In this issue

A xephon PUBLICATION

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.

Printed in England.

Automatic maintenance – hints and tips for manual administration you won't find in the documentation

The IBM DB2 Universal Database V8.2 product for Linux, Unix, and Windows (DB2 UDB) introduced a whole set of automatic features that drive down the administration costs of a DB2 UDB database solution. Among these new features is the ability to set up DB2 UDB to perform automated maintenance for the most routine tasks.

When planning this DB2 UDB release, the IBM development teams interviewed hundreds of DBAs to find out what tasks they needed the most help with from a time-savings perspective. Overwhelmingly, the answers were:

- Back-up
- Table reorganization
- Statistics collection.

First, automated statistics collection was added (with support for intelligent sampling). This feature takes the guesswork out of the when-to-run-it decision by showing the associated resource consumption trade-offs. Furthermore, there are capabilities within DB2 UDB that make the optimizer 'self-learning'; after choosing an access plan, it can decide for itself whether more statistics would be beneficial in the future. For example, perhaps including statistics for a key column would have produced a faster query set response time – DB2 UDB can now figure this out on its own and start to collect the improved set of statistics for you. Now that's smart!

The back-up utility gets a boost of automated capabilities too. As of DB2 UDB V8.2, the back-up utility is self-tuning. I've seen some customers experience literally a ten-fold improvement over their own tuning efforts with previous versions of DB2 UDB. All major vendors offer the ability to schedule a utility at a specific time – that is not new. In DB2

UDB, the back-up utility can now be policy-based: this allows you to tie a back-up to the amount of log space consumed, if you want. (You can easily determine how much log space a set of your transactions consumes.) This relationship allows you to have business-policy-driven back-ups as well as the traditional 'every-Thursday-at-midnight' schedule.

For example, let's look at a typical online florist e-tailer. In 2005, Valentine's Day, 14 February, fell on a Monday. The transaction volume at this florist between Saturday, 12 February and Tuesday, 15 February (for those late people attempting to get out of the 'dog-house') was likely larger than the entire rest of the month. If you had scheduled a back-up to occur every Friday, for example, you could be in for a lot of trouble should a failure occur during this particular week. A DB2 UDB solution would recognize this and spawn as many back-ups as needed to ensure that you, the florist, were not out of policy with respect to the number of transactions that could be lost should a failure occur. On top of this, when DB2 UDB runs the automatic back-up utility, it will throttle itself (if you want). With this feature you set another policy whereby you define how much of a performance impact you're willing to accept while running the utility. For example, you may have a 20% leeway spread built into your Service Level Agreement (SLA), so you're willing to impact the performance of your production system by 15%.

Finally, in DB2 UDB V8.2, table reorganizations can be performed automatically for you, when certain conditions are met, if you like.

All of these automated features can be enabled such that they identify a problem to the DBA, recommend a solution to the DBA, and run the required maintenance, or just run the required maintenance without requiring input from the DBA.

What's more, you can set up offline and online maintenance windows that are tailored to your environments and have DB2 UDB decide what to run and when. For example, let's suppose you had scheduled an online back-up to run between certain

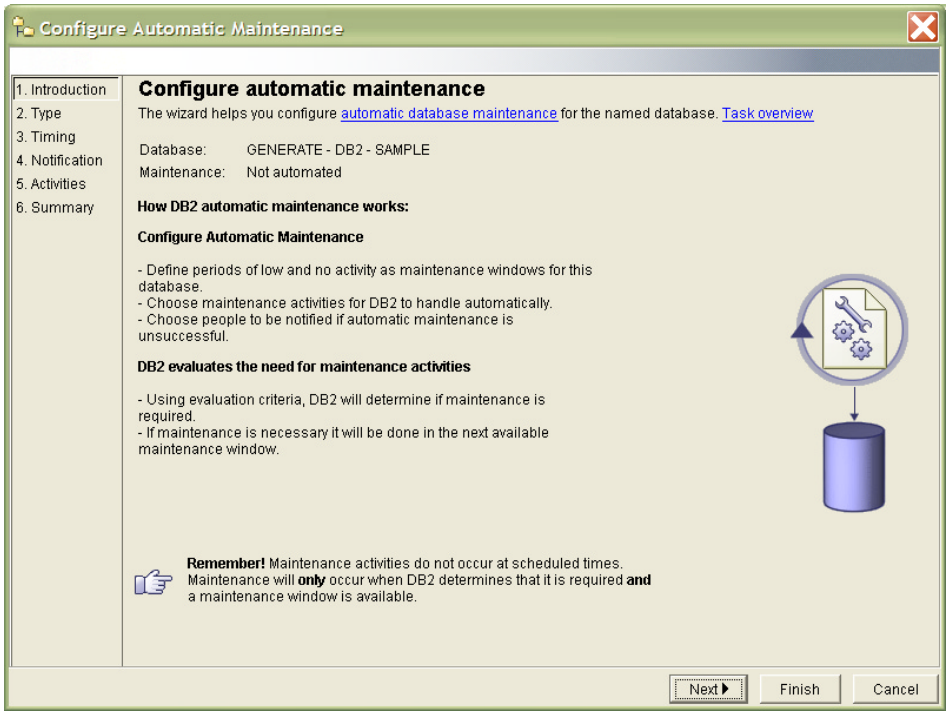


Figure 1: Database maintenance wizard

times. If the length of the back-up would exceed the thresholds you defined, DB2 UDB would run it the first time and alert you to this fact.

You set up a policy for the maintenance of your DB2 UDB server using the Database Maintenance wizard available from the Control Center (see Figure 1).

NOW FOR THE THINGS YOU WON'T FIND IN THE DOCUMENTATION

All of these automated features in DB2 UDB are great, but sometimes you'd like to interact with DB2 UDB without the use of a graphical user interface for many reasons. For example, you are deploying a large number of DB2 UDB servers, or perhaps you sell an application that is built on the DB2 UDB engine.

Table - POLICY

Schema : SYSTOOLS
Creator : PAULZ
Columns : 5

Key	Name	Data type	Length	Nullable
	MED	VARCHAR	128	No
	DECISION	VARCHAR	128	No
	"NAME"	VARCHAR	128	No
	UPDATE_TIME	TIMESTAMP	10	No
	POLICY	BLOB	2097152	Yes

Figure 2: Example schema

All of the information for your maintenance plans is stored in the SYSTOOLS.POLICY table, whose schema looks like the one shown in Figure 2.

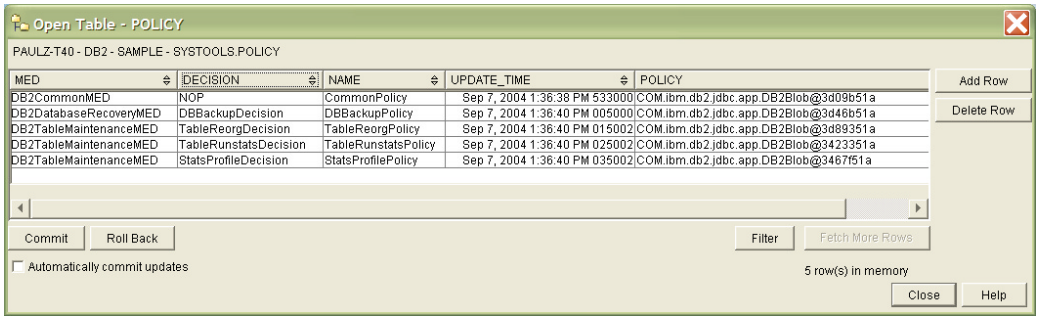


Figure 3: Policy information

Since the policy information is stored as an XML BLOB, I, ironically, use the Control Center to the non-BLOB based information shown in Figure 3. If you want to examine the contents of the XML file, you'll have to extract it from this LOB.

It isn't possible in DB2 UDB (today) to update the policy settings using a Command Line Processor (CLP) or Application Programming Interface (API). The documentation generally guides you to use the graphic wizard (shown previously) to set

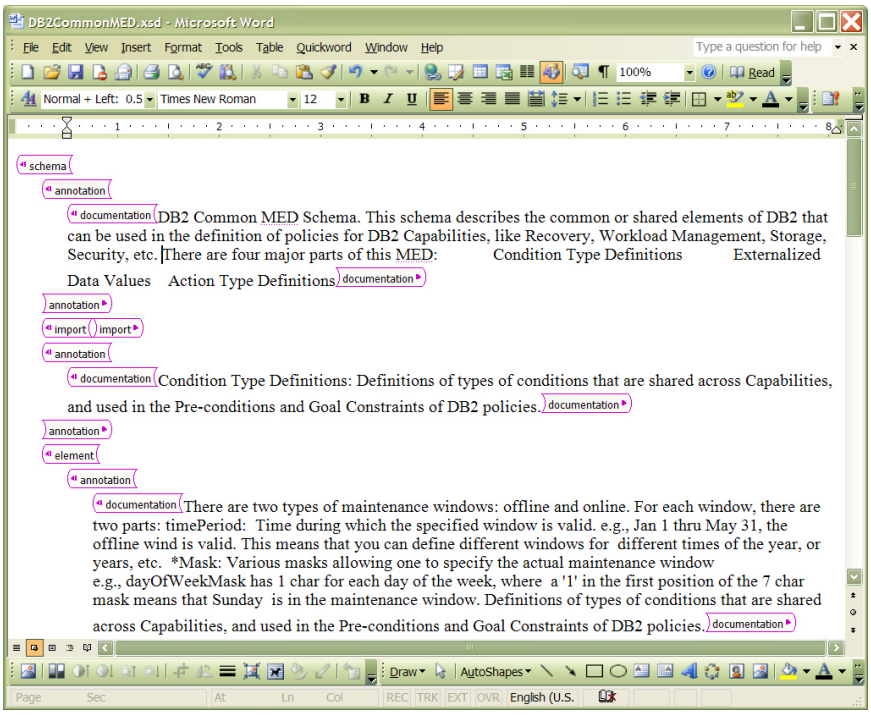


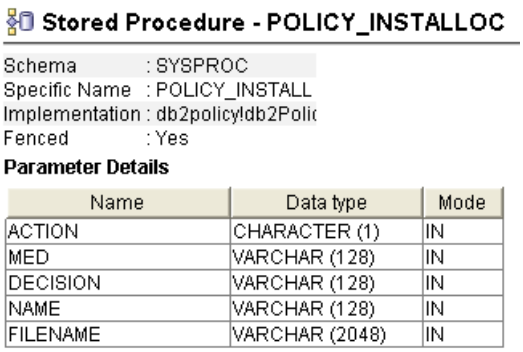
Figure 4: Example DB2CommonMED.xsd

this up. However, you can manually edit or create an XML document that conforms to the DB2 UDB schema for this task, and specify the maintenance plan. After all, this is exactly what DB2 UDB does when you fill out the wizard.

The schema for automatic maintenance plans is called DB2CommonMED.xsd and is located in the <installation path>/SQLLIB/MISC directory. This file is shown in Figure 4 using Microsoft Word.

After you create your own XML document that conforms to this schema, you need to register your maintenance plan with DB2 UDB. You can use the SYSPROC.POLICY_INSTALLOC stored procedure to update the definition.

The definition of the SYSPROC.POLICY_INSTALLOC stored procedure is shown in Figure 5.



Stored Procedure - POLICY_INSTALLOC

Schema : SYSPROC
Specific Name : POLICY_INSTALL
Implementation : db2policy!db2Polic
Fenced : Yes

Parameter Details

Name	Data type	Mode
ACTION	CHARACTER (1)	IN
MED	VARCHAR (128)	IN
DECISION	VARCHAR (128)	IN
NAME	VARCHAR (128)	IN
FILENAME	VARCHAR (2048)	IN

Figure 5: Definition of SYSPROC.POLICY_INSTALLOC

For example, to call this stored procedure to update the current maintenance policy you would enter the following command (specified by the U flag). If a maintenance policy does not exist, you would use the C flag to create one:

```
db2 call SYSPROC.POLICY_INSTALLOC    ('U','DB2CommonMED','NOP',",'<path
name>')
```

WRAPPING IT UP

Should you set up maintenance policies solely using this approach? My advice is no. Personally, I like using GUI controls and wizards to help me script a task, and then I save that task if I need to use it again. However, if you are a DBA responsible for a large rollout of DB2 UDB servers, or perhaps an Independent Software Vendor (ISV) who would like to ship DB2 UDB configured out-of-the-box (even for maintenance in consideration of your application) then this may be a convenient way to do that. This approach requires some educational work. (I encourage you to explore the XML documents and schemas and try changing your maintenance policies in a test environment to get the hang of things.) But once you get used to these features, they can become real time-savers and help solve problems, sometimes even before you know you have them.

The automated features I detailed in this article are just the beginning. DB2 UDB has other automated features, also sometimes called autonomic features, for managing space, deadlocks, memory, and more. You can learn all about these features by reading the *DB2 UDB: The Autonomic Computing Advantage* e-book at www.db2mag.com/db2auto/index.jhtml.

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

DB2 UDB for LUW 8.2: the recover db command

This article looks at the recover database command in DB2 UDB for LUW 8.1 FP8 (V8.2.1).

Consider the scenario – we want to perform a database point-in-time recovery – what do we do? Up to now what we had to do was restore the database, by selecting an appropriate back-up image and using the restore command, and then roll forward through the logs to our point in time using the rollforward command. This was a two-step operation. What's new in Version 8.1 is that we can now perform the database recovery and roll forward of the logs to the point in time in a single step, using the new recover db command.

So what's the difference between the restore db command and the new recover db command? Let's look at the options for both commands:

```
>db2 ? restore db
RESTORE DATABASE source-database-alias { restore-options | CONTINUE |
ABORT }

restore-options:
  [USER username [USING password]] [TABLESPACE [ONLINE] |
  TABLESPACE (tblspace-name [ {,tblspace-name} ... ]) [ONLINE] |
  HISTORY FILE [ONLINE] | LOGS [ONLINE] | COMPRESSION LIBRARY [ONLINE]]
```

```
  [INCREMENTAL [AUTOMATIC | ABORT]] [USE {TSM | XBSA} [OPEN num-sess
SESSIONS]
  [OPTIONS {options-string | options-filename}] |
  FROM dir/dev [{,dir/dev} ... ] | LOAD shared-lib [OPEN num-sess
SESSIONS]
  [OPTIONS {options-string | options-filename}]] [TAKEN AT date-time]
  [TO target-directory] [INTO target-database-alias] [LOGTARGET
directory]
  [NEWLOGPATH directory] [WITH num-buff BUFFERS] [BUFFER buffer-size]
  [DLREPORT file-name] [REPLACE HISTORY FILE] [REPLACE EXISTING]
[REDIRECT]
  [PARALLELISM n] [COMPRLIB lib-name] [COMPROPTS options-string]
  [WITHOUT ROLLING FORWARD] [WITHOUT DATALINK] [WITHOUT PROMPTING]

>db2 ? recover db
RECOVER DATABASE database-alias [TO {isotime
[USING LOCAL TIME | USING GMT TIME] [ON ALL DBPARTITIONNUMS] |
END OF LOGS [On-DbPartitionNum-Clause]}] [USER username
[USING password]] [USING HISTORY FILE (history-file [{, history-file
ON DBPARTITIONNUM db-partition-number} ... ])] [OVERFLOW LOG PATH
(log-directory [{,log-directory ON DBPARTITIONNUM db-partition-number}
... ])]
[COMPRLIB lib-name] [COMPROPTS options-string]
```

You can see that the restore db command has more options than the recover db command. All the options are described in detail in the *Command Reference* (SC09-4828-01). Here are a couple of questions that highlight the appropriate command to use in particular situations..

Can I restore a table space as opposed to the whole database using the recover db command? No, the recover db command does not have a table space option – if you want to recover just a table you have to use the restore db command.

Can I use the recover db command to make a copy of the database with a new name? No, the recover db command does not have a 'to database name' option – if you want to restore the database to a different name you have to use the restore db command.

Let's now look at the recover db command in detail using the following scenario. The following was run on a Windows 2000 Professional system running DB2 UDB V8.2.1 using the db2admin userid. I have a database called HMDB with archive

logging switched on. I make some updates to the database, note a point in time, make some more updates, and then decide to recover to the point in time after the first set of updates – this is all detailed below:

- Create a database called HMDB.
- Switch on archive logging.
- Take an offline back-up.
- Create a table called TB01.
- Insert two rows into TB01.
- Set a point in time (point 1).
- Insert one row into TB01.
- Select from the table TB01.
- Recover the database to point 1.
- Select from the table TB01.

All the commands are listed below:

```
>db2 create db hmdb
>db2 update db cfg for hmdb using logretain on
>db2 backup db hmdb to c:\db2_backups

>db2 connect to hmdb user db2admin using db2admin

>db2 create table TBØ1 (id int, name char(1Ø))
>db2 insert into TBØ1 values (1,'Helen')
>db2 insert into TBØ1 values (2,'Chantal')
>db2 select current timestamp from sysibm.sysdummy1

1
--------------------------
2ØØ5-Ø2-12-14.29.45.294ØØ1

>db2 insert into TBØ1 values (3,'Anita')

>db2 select * from tbØ1

ID          NAME
----------- ----------
          1 Helen
```

```
          2 Chantal
          3 Anita
```

We now have three rows in our table – but the third insert was a mistake and we now want to recover the database to point-in-time 1. We can cut and paste the timestamp from the select current timestamp command into the recover db command as follows (note the format of the timestamp):

```
>db2 recover db hmdb to 2005-02-12-14.29.45 user db2admin using db2admin
```

Now if we connect to the database and select from table TB01 we see only two rows:

```
>db2 connect to hmdb user db2admin using db2admin

>db2 select * from tb01

ID          NAME
----------- ----------
          1 Helen
          2 Chantal
```

You can see that we recovered back to when the table had only two rows in it, which is what we wanted. This was a very simple example, but I hope I have shown how to use the recover db command. I could still have used the restore db and rollforward commands, but the recover db command combines both functions and makes the recovery much easier.

C Leonard
Freelance Consultant (UK)

Guidelines for coding efficient SQL

Well begun is half done – Aristotle

This article is for readers who already know SQL but, at least sometimes, need to find ways to make SQL run faster.

Traditionally, the same people who write the SQL in the first place, the application developers, do most of the tuning. I hope this article helps developers solve their own tuning problems, especially the most common types. The main problem with SQL tuning is finding the optimum path to the data. (The path to the data is known as the execution plan.) This optimum path is virtually independent of the database, and most of the SQL guidelines mentioned in this article cover a vendor-independent solution to that problem.

WHY TUNE SQL?

Let's describe an application that allows users or possibly another application to enter and manipulate data that your organization stores in a relational database. On the output data, it performs actions such as addition, multiplication, counting, averaging, sorting, and formatting – operations like those you would expect to see in a business spreadsheet. The work the application must do after it gets data out of the database, or before it puts data into the database, is modest by modern computing standards, because the data volumes handled outside of the database are small. Even if the vast number of end users leads to high loads, you can generally throw hardware at the application load.

On the other hand the database behind a business application often examines millions of rows in the database just to return the few rows that satisfy an application query, and this inefficiency can completely dominate the overall system load and performance.

Furthermore, while you might easily add application servers, it is usually much harder to put multiple database servers to work on the same consistent set of business data for the same application, so throughput limits on the database server are much more critical. It is imperative to make your system fit your business volumes, not the other way around.

Improvements to SQL performance tend to be the safest

changes you can make to an application and they help both performance and throughput, with no hardware cost or minimal cost at worst (in the case of added indexes, which require disk space).

Performance and throughput are related, but not identical. For example, on a well-configured system with some idle processors (CPUs), the addition of CPUs might increase throughput capacity but would have little effect on performance since most processes cannot use more than a single CPU at a time. Faster CPUs help both throughput and performance of a CPU-intensive application. Getting faster SQL is much like getting faster CPUs, without the additional hardware cost. Performance problems translate to lost productivity because end users waste time waiting for the system. You can eliminate the bottleneck (for example add CPUs) if you are not already at the system limit, or you can tune the application, especially its SQL.

LIST OF SQL GUIDELINES

Performance: power up your SQL

1 SQL should always list the named columns to be returned to the program. SELECT * should never be used. The program should ask for the return of only the columns actually needed by the program. Returning extra unnecessary columns uses more resources and takes more time.

2 Limit the data selected in SQL. Return the minimum number of columns and rows needed by your application program by making efficient use of the WHERE (SQL predicate) clause. It is almost always more efficient to allow DB2 to use the WHERE clause to limit the data returned.

3 The host variable in the SQL should always be defined as the same type and length as the target DB2 table column.

If not, performance will be negatively impacted because DB2 will not use an index. For example, comparing a column defined as CHAR(6) to a field which is CHAR(4) or CHAR(7) will cause a data conversion and will also cause DB2 to scan the data rather than use an index. This should be avoided at all costs.

4 The ORDER BY phrase in SQL should be used with all cursor processing where sequence is important, even if the desired sequence is the clustering sequence. It is not guaranteed that DB2 will return the data in clustering sequence without the ORDER BY. Order only those columns that are absolutely necessary in order to improve efficiency.

5 Singleton SELECT versus the cursor – to return a single row, an application program can use a cursor or a singleton SELECT. A cursor requires an OPEN, FETCH, and CLOSE to retrieve one row, whereas a singleton SELECT requires only SELECT... INTO. Usually, the singleton SELECT outperforms the cursor. When the selected row must be updated after it is retrieved, however, using a cursor with the FOR UPDATE OF clause is recommended over a singleton SELECT. The FOR UPDATE OF clause ensures the integrity of the data in the row because it causes DB2 to hold an exclusive lock on the page containing the row to be updated. The singleton select provides ‘shared’ access to the data page.

6 Use FOR FETCH ONLY. When a SELECT statement will be used only for retrieval, use the FOR FETCH ONLY clause.

7 Avoid using DISTINCT. The DISTINCT verb removes duplicate rows from an answer set. If duplicates will not cause a problem, do not code distinct, because it adds overhead by invoking a sort to remove the duplicates.

8 JOINS should be used where possible to enhance performance. It is preferable to limit a JOIN to a maximum

of five tables, but exceptions may occur. If you want to join more than five tables, please obtain clearance from the DBA group. Use joins instead of sub-queries. Joins will give the DB2 optimizer more options for data access than a sub-query. However, always test both to verify which is performing better for the particular query because data volume could affect the run-time considerably.

9 On an exception basis, you may specify the number of rows to be returned. When you are coding a cursor to fetch a predictable number of rows, consider specifying the number of rows to be retrieved in the OPTIMIZE FOR *n* ROWS clause of the CURSOR. This gives DB2 the opportunity to select the optimal access path for the statement based on actual use. Again, this should be used very infrequently and on an exception basis.

10 Try to sort only on indexed columns. When using ORDER BY, GROUP BY, DISTINCT, and UNION, it is best to use only indexed columns. If the sequence, including ascending and descending, of the columns in your ORDER BY, GROUP BY, etc, are the same as your index, you'll have a better chance to avoid a sort.

11 Avoid using the substring function in an SQL predicate if there are alternative methods.

12 Use SQL built-in functions. It is more efficient to use the SQL built-in functions (AVG, COUNT, MAX, MIN, SUM) than let the application program perform these functions (before using these, be sure to zero the variable you select into and check for SQLCODE 0, -305, or other). Also code for null values because these functions can return a null. When using in a sub-select on an indexed column, use a VALUE clause of the datatype of the column as a default value. This enables DB2 to use an index, if possible. If you do not do this, the column will not be indexable due to the fact that a null value can be returned.

13 Avoid SQL scalar functions if you have alternative methods.

Avoid the use of the SQL Scalar functions used for data type conversions, character string manipulation, and date/ time conversions (ie INTEGER, DECIMAL, HEX, SUBST, etc).

14 Columns that are updated infrequently should be placed at the beginning of a table while columns that are updated frequently should be placed at the end of a table. This is an efficiency consideration related to how DB2 records changes on the DB2 log.

15 In general, VARCHAR columns should be the last columns in a table. Exceptions can be made when there are other columns in the table that are updated more frequently. In that case, the high update columns should be at the end of the table after the VARCHAR.

16 Use BETWEEN rather than <= and >=. BETWEEN allows the optimizer to select a more efficient access path.

17 When performing large batch type processes, the driving input data should be processed in the same sequence as the other tables to be accessed. This may allow DB2 to invoke sequential pre-fetch, which greatly enhances performance.

18 When adding large volumes of data to a table, it is faster to use a database utility. There is a LOAD REPLACE utility in which the data replaces the existing data and a LOAD RESUME utility in which the data is added to the existing data. You are encouraged to use some form of LOAD utility for this type of process. It is advisable to get input on the set-up and running of LOAD utilities from the DBA staff. Do not use the LOG YES parameter with the LOAD utilities; use LOG NO for efficiency purposes.

19 When accessing a table, specify all the index data known to the program. This allows DB2 to maximize the use of index processing. DB2 usually performs more efficiently when it can satisfy a request using an existing index rather than no index. Design all SQL statements to take advantage

of indexes. If a table has only multicolumn indexes, try to specify the high-level column in the WHERE clause of your query. This results in an index scan with at least one matching column.

20 DB2 locking must be considered when developing a program. It is possible to exceed the maximum allowable number of locks. If you do, DB2 will abend the transaction. Commits release locks and should be used.

21 There is overhead associated with commits. Committing more frequently could impact execution time. But there are instances where a large number of updates occur within a single unit of work. When this occurs, the commit frequency may need to be adjusted down to a lower level.

22 Use IN instead of LIKE. If you know that only a certain number of occurrences exist, using IN with the specific list is more efficient than using LIKE. The functionality of LIKE can be imitated using a range of values. For example, if you want to retrieve all employees with a last name starting with M, use BETWEEN 'maaaaaaaaaaaaaa' and 'mzzzzzzzzzzzzzz' instead of LIKE 'm%'.

23 Avoid using NOT (except with EXISTS). NOT should be used only as an alternative to very complex predicates.

24 Avoid using columns in SELECT clauses if that column value is already known in the application program.

```
SELECT C1, C2, C3
INTO: HV_C1,  :HV_C2,  :HV_C3
from T1
WHERE  C1 = :HV_C1
```

25 Avoid arithmetical expressions in SQL predicates by moving the arithmetic to the host programming language.

Use:

COBOL:

```
COMPUTE  HV1  = HV_C3 * 1.1
```

SQL:

```
EXEC SQL
     SELECT C1, C2  INTO :HV_C1, :HV_C2  FROM T1
     WHERE C3  >  :HV1
END-EXEC.
```

Instead of:

```
SELECT C1, C2 FROM T1 WHERE C3  >  :HV_C3 * 1.1
```

26 Use UNION ALL instead of UNION when you know that SELECT statements will not return duplicates.

27 When you code a subquery using negation logic, use NOT EXISTS instead of NOT IN, if possible, to increase the efficiency of your SQL statement. When you use NOT EXISTS, DB2 must verify only non-existence, which can reduce processing time significantly. With the NOT IN predicate, DB2 must materialize the complete result set.

28 A JOIN can be more efficient than a correlated subquery or a subquery:

```
SELECT ENO,ENAME
INTO :HV_ENO, :HV_ENAME
FROM EMP JOIN PROJ
ON WORKDEPT = DEPTNO
WHERE EMPNO  = RESPEMP
```

is more efficient than:

```
SELECT ENO, ENAME
INTO :HV_ENO, :HV_ENAME
FROM EMP
WHERE  WORKDEPT =   (SELECT DEPTNO
FROM PROJ
WHERE
RESPEMP =  EMP.EMPNO)
```

29 Always try to use JOIN on indexed (clustered) columns while joining two tables using SELECT SQL.

Correctness: leave no room for bugs

30 On an INSERT, always specify all the columns in the DB2 table. This will ensure that each column contains the

expected value without making assumptions about which column will be populated.

31 The DB2 return code must always be interrogated and handled properly after each call to DB2. A successful return should always be checked to allow the processing to proceed. In some instances, all other codes should be reported as an error via the standard abort routine. In other instances, a duplicate row may be permissible. If so, the program should do a positive check for a duplicate row and allow processing to continue without any error processing. Checking for a not found condition may also be an error in some instances and a valid situation in others. The program ID, section number, error number, table name, DB2 function, SQL code, and SQLCA information should be moved to the error routine linkage.

32 The easiest and best way to ensure data type consistency is to use the DCLGEN fields whenever possible. Use the DCLGEN area for both the WHERE predicates and INTO clauses whenever possible.

33 In general, any program that updates a DB2 table should have checkpoint restart code. This is true even if the updates were small in number because even low update programs can be holding many read locks.

34 In general, any program that updates DB2 tables should not also produce a sequential file output. This is because there is no way to make the sequential file output correct on a restart. One solution is to write the sequential file to a table and then unload the table. Another solution is to redesign the process so it is more manageable.

35 Current date and time should be obtained from DB2. Current date and time are set automatically on an insert and only need to be reset on an update.

Readability and maintainability: write neat SQL

36 Hard-coded values should not be used in SQL for better

flexibility and maintainability. A host variable should be used instead.

37 Avoid selecting or fetching INTO a group-level host variable structure. Using individual host variables instead of host structures makes programs easier to understand, easier to debug, and easier to maintain:

```
Ø1 WS-ROW.
      Ø5  HV_C1          PIC X(4).
      Ø5  HV_C2          PIC X(2).

FETCH CUR1 INTO :HV_C1, :HV_C2
```

instead of:

```
FETCH C1  INTO :WS-ROW
```

38 SELECT, FROM, WHERE, and other reserved words should be right aligned against the SELECT statement.

39 Operators should be aligned with ample white space between their variables. This allows developers to scan down the line for specific conditions.

40 Tables should be listed one per line.

41 The correlation variable should be short and should help identify the original table.

42 Commas (to continue a clause) should be placed at the beginning of the next line.

43 Always add a comment to describe the SQL statement. This will help developers to maintain the code.

44 You should always provide a comment when special 'tricks' are used to influence the optimizer.

Restrictions and reminders: always remember the rules of the game

45 If a select clause has COLUMN functions and columns not in COLUMN functions, ALL columns not in COLUMN functions must be included in the GROUP BY clause.

46 With UNION ALL, every query in the stack must return the same number of columns and the data types of the *n*th column of each query in the stack must be compatible.

47 Only one ORDER BY clause is allowed in a UNIONed query.

48 UNION ALL does not eliminate duplicates and it performs better than UNION. Use UNION ALL if the queries cannot return duplicate rows.

49 Column functions may be specified only in SELECT and HAVING and column functions cannot be nested.

50 The number of values a subquery can return must be compatible with the operator in the outer select.

SUMMARY

The biggest factor in the performance of business application is the speed of the SQL. The SQL that most affects the load on a system and the productivity of its end users can usually be improved by a large factor. The most common solution to an SQL tuning problem is a prescription for some combination of changes to the database and more often, changes to the SQL itself. These changes allow the formerly slow statement to run faster, with no changes in functionality and no change in the calling application. This common solution is especially attractive, because it is usually simple and it rarely has unwanted side effects.

T S Laxminarayan
System Programmer
Tata Consultancy Services Ltd (India)

Move data across DB2 subsystems to another LPAR – part 2

This month we conclude the code for moving data across different databases and different platforms.

PULR04: PANEL

```
)Attr Default(%+_)
   ( type(text   ) intens(high) hilite(reverse)
   ] type(text   ) intens(high) hilite(reverse) color(white)
   / type(text   ) intens(high) hilite(reverse) color(yellow)
   [ type(text   ) intens(high) hilite(reverse) color(green)
   + type(text   ) intens(low )
   _ type( input) intens(high) caps(on ) just(left )
   ^ type(output) intens(low ) caps(off) just(asis )
   ~ type(output) intens(low ) caps(off) just(asis ) JUST(RIGHT)
)Body
/ Unload Utility - Selection Result +
+
+Command ===>_zcmd                                              +Scroll
===>_amt +
+
+Press[Enter+to have this service continue.
+Press[End  +to respecify your  PARAMETERS.
+
]Dbname  ]Tsname  ]Creator ]Table             ]         Card]Runstat
]Columns+
+
)Model
^z       ^z       ^z       ^z                 ~z            ^z
~z      +
)Init
  .ZVARS = '(HDB, HTS, HCR, HTB, HCO, HRU, HCC)'
  &amt = PAGE
  if (&si='TS')
     &hcr = &z
     &htb = &z
     &hco = &z
     &hru = &z
     &hcc = &z
)Reinit
)Proc
)End
```

PULR0IC: PANEL

```
)Attr Default(%+_)
   ! type(text)   intens(high) caps(on ) color(yellow)
   $ type(output) intens(high) caps(off) color(yellow) hilite(reverse)
   ? type(text)   intens(high) caps(on ) color(red) hilite(uscore)
   # type(text)   intens(high) caps(off) hilite(reverse)
   } type(text)   intens(high) caps(off) color(white)
   ] type( input) intens(high) caps(on ) just(left ) pad('-')
   { type(output) intens(low ) caps(off) color(white) hilite(reverse)
   ^ type(output) intens(low ) caps(off) just(asis ) color(turquoise)
)Body  Expand(//)
?Select IC for Unload$dbts                                    +
%Command ===>_zcmd                                          / /%Scroll
===>_amt +
!Select with an 'S', press ENTER, or press PF3 to Return
+-------------------------------------------------------------------------
--------
+Valid sel:!S+Select
+-------------------------------------------------------------------------
--------
                      #Ic+
+
#S#Icdate    #Ictime  #ty#Num#Devt#Dsname
+
)Model
]z^z         ^z       ^z ^z  ^z   ^z
+
)Init
  .ZVARS = '(sel icd ict ity idsn medi dsname)'
  &amt = PAGE
  &sel = ''
)Reinit
)Proc
  IF (&sel = '')
     &zedsmsg='Warning! See PF1 message'
     &zedlmsg='You must select image copy dataset'
     .msg = isrz001
)End
```

PUNMES: PANEL

```
)ATTR DEFAULT(%+_)
| TYPE (TEXT)   INTENS(LOW)  COLOR(WHITE)
@ TYPE (TEXT)   INTENS(HIGH) COLOR(RED)   CAPS(OFF) HILITE(REVERSE)
! TYPE (INPUT)  INTENS(NON)  COLOR(GREEN) CAPS(ON)  JUST(LEFT)
# TYPE (OUTPUT) INTENS(LOW)  COLOR(GREEN) CAPS(OFF)
)BODY DEFAULT(%~\) WINDOW(60,8)
!ZCMD +            @ Message display !AMT  |
```

```
|-------------------------------------------------------
)MODEL CLEAR(MESSG)
#Z                                        +
)INIT
     .ZVARS = '(MESSG)'
)REINIT
)PROC
  IF (.PFKEY = PFØ3) &PF3 = EXIT
  IF (&ZCMD=END)
     &COMMAND = CANCEL
)END
```

ULUSR: SKELETON

```
)TBA 72
)CM  ------------------------------------------------------------------
)CM Skeleton to generate jcl for unload - load utility                --
)CM  ------------------------------------------------------------------
//&user.X JOB (12ØØ-12Ø5-ØZ),'&option',
//             NOTIFY=&user,REGION=4M,
//             CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//*  ****************************************************************
//*    UNLOAD - LOAD UTILITY
//*    GENERATION DATE AND TIME : &date AT: &time
//*
//*    UNLOAD - WAS RUN WITH THE FOLLOWING PARAMETERS:
//*    ---------  ----------------------
//*    SSID     :  &db2
//*    Source   :  &si &sid
//*    Creator  :  &crecm
//*    Name     :  &tabcm
//*    Tsname   :  &tsncm
//*    Dbname   :  &dbncm
//*    CCSID    :  &ccsid
//*    Nosubs   :  &nos
//*    Nopad    :  &npd
//*    Float    :  &flo
//*    Maxerr   :  &mer
//*    Shrlevel :  &shr
//*    Isolation:  &ur
//*    Load     :  &loa
//*    Tosystem :  &sysr
//*    Selection:  &wse
)SEL &loa = YES
//*    Loadmeth :  &meth
//*    Log      :  &log
//*    Runstats :  &rus
)ENDSEL
//*  ****************************************************************
```

```
//*  NUM DBNAME   TSNAME   CREATOR  TABLE NAME
//*  -- -------- -------- -------- ------------------
)DOT "TLIST"
//* &detail
)ENDDOT
//*-------------------------------------------------
//* STEP 1: DELETE WORKING DATASETS                 */
//*-------------------------------------------------
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN  DD  *
  DELETE DB2MB.UNLOAD.TABLE.*
  DELETE DB2MB.PUNCH.TABLE.*
  SET MAXCC=Ø
)DOT "TLIST"
//*-------------------------------------------------
//* STEP 2: UNLOAD &hcr..&htb   LPAR: MVSLJ
//*-------------------------------------------------
//UNLO&ntb    EXEC DSNUPROC,SYSTEM=&db2,
//         UID='&user..UNLOAD&ntb',UTPROC='',COND=(4,LT)
)SEL &poz = C
//SYSREC   DD DSN=DB2MB.UNLOAD.TABLE.TB&ntb,
//         STORCLAS=NONSMS,
//         UNIT=339Ø,VOL=SER=MVSDB1,
//         DISP=(NEW,CATLG,CATLG),
//         SPACE=(TRK,(&pri,&sec),RLSE)
//SYSPUNCH DD DSN=DB2MB.PUNCH.TABLE.TB&ntb,
//         STORCLAS=NONSMS,
//         UNIT=339Ø,VOL=SER=MVSDB1,
//         DISP=(NEW,CATLG,CATLG),
//         SPACE=(TRK,(1,1),RLSE)
)ENDSEL
//SYSPRINT DD SYSOUT=*
//SYSIN    DD  *
)SEL &poz = L
  LISTDEF UNLDLIST
          INCLUDE TABLESPACE &hdb..&hts
)ENDSEL
)SEL &poz = T OR &poz = L
  TEMPLATE ULDDDN
     DSN(DB2MB.UNLOAD.TABLE.TB&ntb)
     UNIT(SYSALLDA) SPACE(1ØØ,1ØØ) CYL DISP(NEW,CATLG,CATLG)
     VOLUMES(MVSDB1)
  TEMPLATE PNHDDN
     DSN(DB2MB.PUNCH.TABLE.TB&ntb)
     UNIT(SYSALLDA) SPACE(1,1) CYL DISP(NEW,CATLG,CATLG)
     VOLUMES(MVSDB1)
)ENDSEL
)SEL &poz = C OR &poz = T
```

```
  UNLOAD TABLESPACE &hdb..&hts
)ENDSEL
)SEL &poz = L
  UNLOAD LIST UNLDLIST
)ENDSEL
)SEL &si = IC
         FROMCOPY &icdsn
         FROMVOLUME CATALOG
)ENDSEL
)SEL &poz = T OR &poz = L
         PUNCHDDN PNHDDN  UNLDDN ULDDDN
)ENDSEL
)SEL &spec1 = 1
  &unloads
)ENDSEL
)SEL &spec2 = 1
  &shrlevel
)ENDSEL
)SEL &poz = C OR &poz = T
  FROM TABLE &hcr..&htb
)ENDSEL
)SEL &c1 EQ 1 OR &c2 EQ 1
      (
)ENDSEL
)SEL &c1 = 1
       &colu1
)ENDSEL
)SEL &c2 = 1
       &colu2
)ENDSEL
)SEL &c1 EQ 1 OR &c2 EQ 1
      )
)ENDSEL
)SEL &i1 EQ 1 OR &i2 EQ 1 OR &i3 EQ 1 OR &i4 EQ 1 OR &i5 EQ 1
  WHEN(
)ENDSEL
)SEL &i1 = 1
       &item1
)ENDSEL
)SEL &i2 = 2
       &item2
)ENDSEL
)SEL &i3 = 3
       &item3
)ENDSEL
)SEL &i4 = 4
       &item4
)ENDSEL
)SEL &i5 = 5
```

```
       &item5
)ENDSEL
)SEL &i1 EQ 1 OR &i2 EQ 1 OR &i3 EQ 1 OR &i4 EQ 1 OR &i5 EQ 1
       )
)ENDSEL
//*
)SEL &loa = YES
//*-------------------------------------------------
//* STEP 3: EDIT PUNCH DATASET
//*-------------------------------------------------
//EDIT     EXEC PGM=IKJEFT01,COND=(4,LT)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
  EDIT 'DB2MB.PUNCH.TABLE.TB&ntb' DATA NONUM
)SEL &meth = REPLACE
  TOP
  CHANGE * 99999 'RESUME YES' 'REPLACE' ALL
  TOP
  CHANGE * 99999 'LOG NO' 'LOG NO NOCOPYPEND ENFORCE NO' ALL
)ENDSEL
)SEL &rus = YES AND &meth = REPLACE
  TOP
  FIND 'LOAD DATA'
  INSERT  STATISTICS TABLE(ALL) INDEX(ALL)
  INSERT  REPORT YES UPDATE(ALL)
)ENDSEL
  END SAVE
//*
//*-------------------------------------------------
//* STEP 4: TRANSMIT JCL IN ANOTHER LPAR
//*-------------------------------------------------
//JCLGEN    EXEC  PGM=IKJEFT01,DYNAMNBR=20,COND=(7,LT),REGION=4096K
//STEPLIB   DD    DSN=CEE.SCEERUN,DISP=SHR
//          DD    DSN=DSN710.SDSNLOAD,DISP=SHR
//          DD    DSN=SKUPNI.BATCH.LOADLIB.Y,DISP=SHR
//SYSTSPRT  DD    SYSOUT=X
//SYSPRINT  DD    SYSOUT=X
//LISTA     DD    SYSOUT=(A,INTRDR),DCB=(RECFM=F,BLKSIZE=80)
//SYSUDUMP  DD    DUMMY
//SYSTSIN   DD    *
 DSN SYSTEM(DSNY)
 RUN PROGRAM(PXMIT)  PLAN(PXMIT)
 END
//*-------------------------------------------------
//*---- MOVE ON LPAR: MVSMB
//*---- LOAD &hcr..&htb
//*-------------------------------------------------
//IN        DD    *
#//&user.X JOB (1200-1205-00),CLASS=A,
```

```
#//             MSGCLASS=X,NOTIFY=&user,
#//             MSGLEVEL=(1,1),USER=,REGION=4M
#/*XMIT  MVSMB
#//&user.X JOB (12ØØ-12Ø5-ØØ),CLASS=A,
#//             MSGCLASS=X,NOTIFY=&user,
#//             MSGLEVEL=(1,1),USER=SYSADM,REGION=4M
#//LOAD&ntb     EXEC DSNUPROC,SYSTEM=&sysr,
#//          UID='&user..LOAD&ntb',UTPROC='',COND=(4,LT)
#//SYSREC    DD DSN=DB2MB.UNLOAD.TABLE.TB&ntb,DISP=SHR
#//SYSUT1    DD DSN=SYSADM.SYSUT1.TABLE.TB&ntb,
#//          DISP=(MOD,DELETE,CATLG),
#//          UNIT=SYSDA,SPACE=(CYL,(1ØØ,1ØØ),RLSE)
#//SYSERR    DD DSN=SYSADM.SYSERR.TABLE.TB&ntb,
#//          DISP=(MOD,DELETE,CATLG),
#//          UNIT=SYSDA,SPACE=(CYL,(1Ø,1Ø),RLSE)
#//SYSMAP    DD DSN=SYSADM.SYSMAP.TABLE.TB&ntb,
#//          DISP=(MOD,DELETE,CATLG),
#//          UNIT=SYSDA,SPACE=(CYL,(1Ø,1Ø),RLSE)
#//SORTWKØ1  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø),,,ROUND)
#//SORTWKØ2  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø),,,ROUND)
#//SORTOUT   DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø),,,ROUND)
#//SYSPRINT DD SYSOUT=*
#//SYSIN     DD DSN=DB2MB.PUNCH.TABLE.TB&ntb,DISP=SHR
#//*
#//CHECKP     EXEC  PGM=IKJEFTØ1,DYNAMNBR=2Ø,COND=(7,LT),REGION=4Ø96K
#//STEPLIB    DD    DSN=CEE.SCEERUN.ZOS,DISP=SHR
#//           DD    DSN=DSN71Ø.SDSNLOAD,DISP=SHR
#//           DD    DSN=SKUPNI.BATCH.LOADLIB,DISP=SHR
#//SYSTSPRT   DD    SYSOUT=X
#//SYSPRINT   DD    SYSOUT=X
#//LISTA      DD    DSN=DB2MB.CHECK.TS,DISP=OLD
#//SYSUDUMP   DD    DUMMY
#//SYSTSIN    DD    *
# DSN SYSTEM(&sysr)
# RUN PROGRAM(PTSCP)  PLAN(PTSCP)
# END
#//*
#//*---- CHECK PENDING -------------------------------------
#//REPAIR  EXEC DSNUPROC,SYSTEM=&sysr,COND=(4,LT)
#//STEPLIB  DD DSN=DSN71Ø.SDSNLOAD,DISP=SHR
#//SYSIN    DD DSN=DB2MB.CHECK.TS,DISP=SHR
#//*
)ENDSEL
)ENDDOT
#/*
```

PXMIT: PL/I SOURCE CODE

```
* PROCESS GS,OFFSET,OPT(TIME);
```

```
 XMIT:PROC(PARMS)OPTIONS(MAIN) REORDER;
 /*******************************************************************/
 /* JCL - XMIT PROCES                                               */
 /*******************************************************************/
  DCL PARMS CHAR(100) VAR;
  DCL IN RECORD INPUT ENV(F RECSIZE(80));
  DCL LISTA FILE OUTPUT ENV(F RECSIZE(80));
  DCL VH   CHAR(80);
  DCL VHOD CHAR(80);
  DCL EOF BIT(1) INIT('0'B);
  ON ENDFILE(IN) EOF='1'B;
  EXEC SQL INCLUDE SQLCA;
  PUT SKIP LIST ('***      S T A R T     ***');
  OPEN FILE(IN);
  READ FILE(IN) INTO(VH);
  DO WHILE(^EOF);
     VHOD=SUBSTR(VH,2);
     WRITE FILE(LISTA) FROM(VHOD);
     PUT SKIP LIST (VHOD);
     READ FILE(IN) INTO(VH);
  END;
  CLOSE FILE(IN);
  PUT SKIP LIST ('***      E N D     ***');
 END XMIT;
```

PTSCP: PL/I SOURCE CODE

```
* PROCESS GS,OFFSET,OPT(TIME);
 PTSCP:PROC(PARMS)OPTIONS(MAIN) REORDER;
 /*******************************************************************/
 /* DESCRIPTION: START CHECK PENDING TS IN FORCE MODE               */
 /*******************************************************************/
  DCL PARMS CHAR(100) VAR;
  DCL SYSPRINT    FILE STREAM OUTPUT;
  DCL LISTA FILE OUTPUT ENV(F RECSIZE(80));
  DCL COMMAND     CHAR(80);

  EXEC SQL INCLUDE SQLCA;
  DCL DATETIME BUILTIN;

  EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR SELECT
     '  REPAIR SET TABLESPACE '!!STRIP(DBNAME)!!'.'!!
                        STRIP(NAME)!!' NOCHECKPEND'
  FROM SYSIBM.SYSTABLESPACE
  WHERE STATUS IN ('P','S')
    AND DBNAME LIKE '%'
    AND NAME LIKE '%'
  WITH UR;
  EXEC SQL OPEN C1;
```

```
  EXEC SQL FETCH C1 INTO :COMMAND;

  DO WHILE (SQLCODE=0);
     WRITE FILE(LISTA) FROM(COMMAND);
     EXEC SQL FETCH C1 INTO :COMMAND;
  END;
  EXEC SQL CLOSE C1;
 END PTSCP;
```

Bernard Zver (bernard.zver@informatika.si)
DBA
Informatica (Slovenia)

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

XML is an abbreviation of eXtensible Mark-up Language. It is extensible in that the language itself is a meta language, allowing you to create your own language, in accordance with the needs of your enterprise. You use XML to capture not only the data for your particular application, but also the data structure. XML is not the only interchange format. However, XML has emerged as the accepted standard for data interchange. By adhering to this standard, applications can finally share data without needing to transform data using proprietary formats.

An XML document is considered well written when its syntax is correct, and valid when it respects a document model. However, because XML is a meta language, there are an infinite number of XML formats, and most XML documents should respect a particular document model, which can be defined in one of two ways:

- By a Document Type Definition (DTD).
- By an XML schema.

This article describes an application that presents a graphical model of the hierarchical structure of the DTD and XML document, with the ability to show individually each record from an XML document by simply selecting it from a combo box. It also enables DTD mapping to DB2 tables, assuming that XML documents have a Document Type Definition associated with them, either stored in a separate file or embedded within the same XML file.

An XML Document Type Definition (DTD) is a set of declarations for XML elements and attributes. The DTD defines which elements are used in the XML document, in what order they can be used, and which elements can contain other elements. The location path is a sequence of XML tags that identify an XML element or attribute. The location path identifies the structure of the XML document, indicating the context for the element or attribute.

A common question in the XML community is how to map XML to databases. This application uses a table-based mapping (as opposed to object-relational mapping). The table-based mapping is used as the basis for software that transfers data between XML documents and databases, especially relational databases. There is an obvious mapping between the following XML document and table:

```
<A>
   <B>
      <C>ccc</C>                Table A
      <D>ddd</D>                -------
      <E>eee</E>                 C     D     E
   </B>                        ---   ---   ---
   <B>              <=>       ...   ...   ...
      <C>fff</C>               ccc   ddd   eee
      <D>ggg</D>               fff   ggg   hhh
      <E>hhh</E>               ...   ...   ...
   </B>
</A>
```

The table-based mapping views the document as a single table or a set of tables. The mapping has several disadvantages; primarily, it works with only a very small subset of XML

documents. In addition, it does not preserve physical structure (such as character and entity references, CDATA sections, character encodings, or the stand-alone declaration) or document information (such as the document type or DTD), comments, or processing instructions. The table-based mapping is commonly used by middleware to transfer data between XML documents and relational databases. It is also used in some Web application servers to return result set data as XML.

The application works in the following way: it chooses an XML file document that has a Document Type Definition associated with it. The graphical view of the hierarchical structure, with values from the first record in the document, is displayed. By clicking on the element marked by '*' in its graphical presentation, a dialog panel positioned on the corresponding combo box is opened. The required record is then extracted and the diagram is refreshed with the chosen data. Clicking on the button **Mapping to DB2**, a file with DDL statements for creating all the necessary DB2 tables is generated in the current directory. Column data types are derived by parsing data in the XML file. A screenshot from the application is shown in Figure 1.

The compilation of the Java source code is executed by the following command:

```
javac -deprecation MyFileFilter.java
javac -deprecation XmlDtd.java
```

In order to use file access functions, the following lines should be inserted in the java.policy file:

```
grant {
  ...
  permission java.util.PropertyPermission "user.dir", "read";
  permission java.lang.RuntimePermission "modifyThread";
  permission java.io.FilePermission "<<ALL FILES>>", "read, write,
delete, execute";};
  ...
};
```

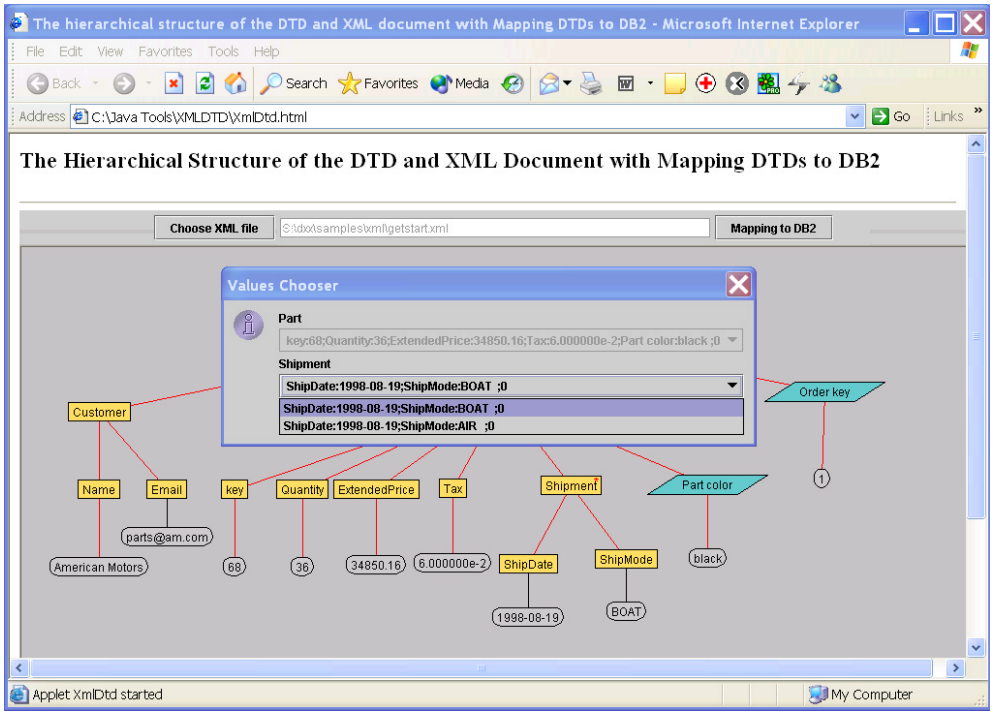


Figure 1: Hierarchical structure of a DTD and XML document

XMLDTD.JAVA

```
/* ********************************************************** */
/* XmlDtd.java                                                */
/*    The hierarchical structure of the DTD and XML document  */
/*    with Mapping DTDs to DB2                                */
/* ********************************************************** */
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.text.*;
import javax.swing.border.*;
import javax.swing.colorchooser.*;
import javax.swing.filechooser.*;
import javax.accessibility.*;
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import java.io.*;

class Node {
    double x;
```

```
    double y;
    double dx;
    double dy;
    String parent;
    String type;
    String lbl;
}

class Edge {
    int from;
    int to;
    double len;
}

class MsgBox extends JDialog implements ActionListener {
  JButton okbtn;
  MsgBox(JFrame frame, String msg) {
    super(frame, "Message");
    getContentPane().setLayout(new BorderLayout());
    getContentPane().add("Center",new JLabel(msg));
    JPanel p = new JPanel();
    p.setLayout(new FlowLayout());
    p.add(okbtn = new JButton("OK"));
    okbtn.addActionListener(this);
    getContentPane().add("South",p);
    Dimension d = getToolkit().getScreenSize();
    setLocation(d.width/3,d.height/3);
    pack();
    setVisible(true);
  }

  public void actionPerformed(ActionEvent ae){
    if(ae.getSource() == okbtn) {
      setVisible(false);
    }
  }
}

class XmlDtdLegend extends JPanel {
    XmlDtd graph;
    public XmlDtdLegend(XmlDtd graph) {
      this.graph = graph;
    }
    final Color edgeColor = Color.black;
    final Color nodeColor = new Color(250, 220, 100);
    final Color attrColor = new Color(100, 200, 200);
    final Color valColor = new Color(200, 200, 200);
    public void paint(Graphics g) {
      Dimension d = getSize();
      g.setColor(getBackground());
```

```
      g.fillRect(0, 0, d.width, d.height);
      FontMetrics fm = g.getFontMetrics();
      int w = fm.stringWidth("ELEMENT") + 10;
      int h = fm.getHeight() + 4;
      g.setColor(Color.black);
      int x = d.width / 8;
      int y = 12;
      g.drawString("LEGEND:", x - (w-10)/2, (y - (h-4)/2) +
fm.getAscent());
      x = d.width / 4;
      g.setColor(nodeColor);
      g.fillRect(x - w/2, y - h / 2, w, h);
      g.setColor(Color.black);
      g.drawRect(x - w/2, y - h / 2, w-1, h-1);
      g.drawString("ELEMENT", x - (w-10)/2, (y - (h-4)/2) +
fm.getAscent());
      x = (d.width / 4) * 2;
      g.setColor(attrColor);
      int[] polyX;
      int[] polyY;
      polyX=new int[4];
      polyY=new int[polyX.length];
      polyX[0]= x + (w / 2);
      polyY[0]= y + (h / 2);
      polyX[1]= x + w;
      polyY[1]= y - (h / 2);
      polyX[2]= x - (w / 2);
      polyY[2]= y - (h / 2);
      polyX[3]= x - w;
      polyY[3]= y + (h / 2);
      g.fillPolygon(polyX, polyY, polyX.length);
      g.setColor(Color.black);
      g.drawPolygon(polyX, polyY, polyX.length);
      g.drawString("ATTRIBUTE", x - (w-10)/2, (y - (h-4)/2) +
fm.getAscent());
      g.setColor(valColor);
      x = (d.width / 4) * 3;
      g.fillRoundRect(x - w/2, y - h / 2, w, h, 20, 20);
      g.setColor(Color.black);
      g.drawRoundRect(x - w/2, y - h / 2, w-1, h-1, 20, 20);
      g.drawString("VALUE", x - (w-10)/2, (y - (h-4)/2) +
fm.getAscent());
      x = (d.width / 4) * 3 + 100;
        g.setColor(Color.red);
      g.drawString("* - repeat (click for choice)", x - (w-10)/2, (y -
(h-4)/2) + fm.getAscent());
      g.setColor(Color.black);
    }
}
```

```
class XmlDtdPanel extends Canvas
    implements Runnable, MouseListener, MouseMotionListener,
ActionListener {
    XmlDtd graph;
    static int nnodes;
    static String selind = "0";
    static Node nodes[] = new Node[300];
    static int nedges;
    static Edge edges[] = new Edge[500];
    Thread mythread;
    int maxdx = -1;
    int maxdy = -1;
    static JComboBox[] cb = new JComboBox[XmlDtd.icb + 1];
    XmlDtdPanel(XmlDtd graph) {
      this.graph = graph;
      addMouseListener(this);
    }

    int findNode(String type, String lbl, String posx, String par) {
      for (int i = 0 ; i < nnodes ; i++) {
          if (nodes[i].lbl.equals(lbl) & lbl.equals(par) &
!type.substring(0, 1).equals("v")) {
            return i;
          }
      }
      return addNode(type, lbl, posx, par);
    }

    int addNode(String type, String lbl, String posx, String par) {
      Node n = new Node();
      int posy = Integer.valueOf(posx.substring(0,2)).intValue();
      int ordno = Integer.valueOf(posx.substring(2)).intValue();
      n.x = 10 + (70 * ordno);
      n.y = 10 + (80 * posy);
      n.parent = par;
      n.type = type;
      n.lbl = lbl;
      nodes[nnodes] = n;
      if ((int)n.x > maxdx) {
        maxdx = (int)n.x;
        }
      if ((int)n.y > maxdy) {
        maxdy = (int)n.y;
        }
      graph.resize(maxdx + 100, maxdy + 100);
      return nnodes++;
    }

    void addEdge(String typefrom, String from, String typeto, String to,
int len, String posx, String posy) {
```

```
  Edge e = new Edge();
  e.from = findNode(typefrom, from, posx, from);
  e.to = findNode(typeto, to, posy, from);
  e.len = len;
  edges[nedges++] = e;
}

public void run() {
    Thread me = Thread.currentThread();
  while (mythread == me) {
      relax();
      if (Math.random() < 0.03) {
        Node n = nodes[(int)(Math.random() * nnodes)];
      }
      try {
        Thread.sleep(100);
      } catch (InterruptedException e) {
        break;
      }
  }
}

synchronized void relax() {
  for (int i = 0 ; i < nedges ; i++) {
      Edge e = edges[i];
      double vx = nodes[e.to].x - nodes[e.from].x;
      double vy = nodes[e.to].y - nodes[e.from].y;
      double len = Math.sqrt(vx * vx + vy * vy);
        len = (len == 0) ? .0001 : len;
      double f = (edges[i].len - len) / (len * 3);
      double dx = f * vx;
      double dy = f * vy;
      nodes[e.to].dx += dx;
      nodes[e.to].dy += dy;
      nodes[e.from].dx += -dx;
      nodes[e.from].dy += -dy;
  }
  for (int i = 0 ; i < nnodes ; i++) {
      Node n1 = nodes[i];
      double dx = 0;
      double dy = 0;
      for (int j = 0 ; j < nnodes ; j++) {
        if (i == j) {
            continue;
        }
        Node n2 = nodes[j];
        double vx = n1.x - n2.x;
        double vy = n1.y - n2.y;
        double len = vx * vx + vy * vy;
        if (len == 0) {
```

```
            dx += Math.random();
            dy += Math.random();
          } else if (len < 100*100) {
            dx += vx / len;
            dy += vy / len;
          }
        }
        double dlen = dx * dx + dy * dy;
        if (dlen > 0) {
          dlen = Math.sqrt(dlen) / 2;
          n1.dx += dx / dlen;
          n1.dy += dy / dlen;
        }
      }
      Dimension d = getSize();
      for (int i = 0 ; i < nnodes ; i++) {
        Node n = nodes[i];
          if (n.x < 0) {
            n.x = 0;
          } else if (n.x > d.width) {
            n.x = d.width;
          }
          if (n.y < 0) {
            n.y = 0;
          } else if (n.y > d.height) {
            n.y = d.height;
          }
        n.dx /= 2;
        n.dy /= 2;
      }
      repaint();
    }

    Node pick;
    Image offscreen;
    Dimension offscreensize;
    Graphics offgraphics;
    final Color selectColor = Color.pink;
    final Color edgeColor = Color.black;
    final Color nodeColor = new Color(250, 220, 100);
    final Color attrColor = new Color(100, 200, 200);
    final Color valColor = new Color(200, 200, 200);
    final Color arcColor1 = Color.black;
    final Color arcColor2 = Color.pink;
    final Color arcColor3 = Color.red;

    public void paintNode(Graphics g, Node n, FontMetrics fm) {
        // back to the original names (only lbl)
      if (n.lbl.indexOf("renxxren") > 0) {
        n.lbl = n.lbl.substring(0, n.lbl.indexOf("renxxren"));
```

```
      }
      int x = (int)n.x;
      int y = (int)n.y;
      g.setColor((n == pick) ? selectColor : nodeColor);
      int w = fm.stringWidth(n.lbl) + 10;
      int h = fm.getHeight() + 4;
      if ((n.type.substring(0, 1).equals("r")) | (n.type.substring(0,
1).equals("e"))) {
            g.fillRect(x - w/2, y - h / 2, w, h);
            g.setColor(Color.black);
            g.drawRect(x - w/2, y - h / 2, w-1, h-1);
      } else {
            if (n.type.substring(0, 1).equals("a")) {
                  g.setColor((n == pick) ? selectColor : attrColor);
                  int[] polyX;
                  int[] polyY;
                  polyX=new int[4];
                  polyY=new int[polyX.length];
                  polyX[0]= x + (w / 2);
                  polyY[0]= y + (h / 2);
                  polyX[1]= x + w;
                  polyY[1]= y - (h / 2);
                  polyX[2]= x - (w / 2);
                  polyY[2]= y - (h / 2);
                  polyX[3]= x - w;
                  polyY[3]= y + (h / 2);
                  g.fillPolygon(polyX, polyY, polyX.length);
                  g.setColor(Color.black);
                  g.drawPolygon(polyX, polyY, polyX.length);
            } else {
                  g.setColor((n == pick) ? selectColor : valColor);
                  g.fillRoundRect(x - w/2, y - h / 2, w, h, 20, 20);
                  g.setColor(Color.black);
                  g.drawRoundRect(x - w/2, y - h / 2, w-1, h-1, 20, 20);
            }
      }
      g.drawString(n.lbl, x - (w-10)/2, (y - (h-4)/2) + fm.getAscent());
      if (n.type.indexOf('*') > 0) {
          g.setColor(Color.red);
        g.drawString("*", x + (w-15)/2, y + (fm.getAscent() / 8));
        g.setColor(Color.black);
        }
    }

    public synchronized void update(Graphics g) {
      Dimension d = getSize();
      if ((offscreen == null) || (d.width != offscreensize.width) ||
(d.height != offscreensize.height)) {
          offscreen = createImage(d.width, d.height);
          offscreensize = d;
```

```
            if (offgraphics != null) {
                offgraphics.dispose();
            }
            offgraphics = offscreen.getGraphics();
            offgraphics.setFont(getFont());
        }
        offgraphics.setColor(getBackground());
        offgraphics.fillRect(0, 0, d.width, d.height);
        for (int i = 0 ; i < nedges ; i++) {
            Edge e = edges[i];
            int x1 = (int)nodes[e.from].x;
            int y1 = (int)nodes[e.from].y;
            int x2 = (int)nodes[e.to].x;
            int y2 = (int)nodes[e.to].y;
            int len = (int)Math.abs(Math.sqrt((x1-x2)*(x1-x2) + (y1-
y2)*(y1-y2)) - e.len);
            offgraphics.setColor((len < 10) ? arcColor1 : (len < 20 ?
arcColor2 : arcColor3)) ;
            offgraphics.drawLine(x1, y1, x2, y2);
        }
        FontMetrics fm = offgraphics.getFontMetrics();
        for (int i = 0 ; i < nnodes ; i++) {
            paintNode(offgraphics, nodes[i], fm);
        }
        g.drawImage(offscreen, 0, 0, null);
    }

    public void OptionPanel(String bound, String selnode) {
      JOptionPane op = new JOptionPane();
      JPanel panel = new JPanel();
      int i, j, k, l, curcb;
      l = -1;
      curcb = 0;
      String str = new String();
      String[] options = {
      "OK",
      "Cancel"
      };
      Object[] message = new Object[2 * (XmlDtd.icb + 1)];
      cb = new JComboBox[XmlDtd.icb + 1];
      for (i = 0; i < XmlDtd.icb + 1; i++) {
        cb[i] = new JComboBox();
      for (j = 0; j < XmlDtd.maxp + 1; j++) {
        str = XmlDtd.matrix[i][j];
          if (i > 0) {
            if (str.substring(str.lastIndexOf(';') +
1).trim().equals(bound)) {
            cb[i].addItem(XmlDtd.matrix[i][j]);
          }
        } else {
```

```
          cb[i].addItem(XmlDtd.matrix[i][j]);
            if (str.substring(str.lastIndexOf(';') +
1).trim().equals(bound)) {
            cb[i].setSelectedIndex(j);
          }
        }
      }
        cb[i].addActionListener(this);
      l++;
        message[l] = new
JLabel(XmlDtd.ncb[i].substring(XmlDtd.ncb[i].indexOf(')') + 1,
XmlDtd.ncb[i].indexOf(':')));
      l++;
      message[l] = cb[i];
      if (XmlDtd.ncb[i].substring(XmlDtd.ncb[i].indexOf(')') + 1,
XmlDtd.ncb[i].indexOf(':')).equals(selnode)) {
        curcb = i;
      }
      }
      for (i = 0; i < curcb; i++) {
        cb[i].setEnabled(false);
      }
      int result = op.showOptionDialog(
      panel,
      message,
      "Values Chooser",
      JOptionPane.DEFAULT_OPTION,
      JOptionPane.INFORMATION_MESSAGE,
      null,
      options,
      cb[curcb]
      );
      switch(result) {
      case 0: // OK
        for (i = 0; i < XmlDtd.icb + 1; i++) {
          String cbval = (String)cb[i].getSelectedItem();
            if (i > 0) {
              selind = cbval.substring(cbval.lastIndexOf(';') +
1).trim();
          }
          for (k = 0; k < nnodes; k++) {
            if ((nodes[k].type.substring(0, 1).equals("v")) &
(cbval.indexOf(nodes[k].parent)) >= 0) {
              int sposp = cbval.indexOf(nodes[k].parent);
              int eposc = cbval.indexOf(';', sposp);
              int possc = cbval.indexOf(':', sposp);
              nodes[k].lbl = cbval.substring(possc + 1, eposc);
              cbval = cbval.substring(0, sposp) + cbval.substring(eposc
+ 1);
           }
```

```
          }
        }
        repaint();
        break;
      case 1: // Cancel
        break;
      default:
        break;
      }
    }

    public void cbrefresh(String str) {
      int i, j, k;
      String s = "Ø";
      boolean fnd = false;
      for (i = Ø; i < XmlDtd.icb + 1; i++) {
      for (j = Ø; j < XmlDtd.maxp + 1; j++) {
        if (XmlDtd.matrix[i][j].equals(str)) {
          fnd = true;
          s = str.substring(str.lastIndexOf(';') + 1).trim();
          break;
        }
      }
      if (fnd) {
        break;
        }
      }
      if (fnd) {
        for (k = i + 1; k < XmlDtd.icb + 1; k++) {
        cb[k].removeAllItems();
        for (j = Ø; j < XmlDtd.maxp + 1; j++) {
          if
(XmlDtd.matrix[k][j].substring(XmlDtd.matrix[k][j].lastIndexOf(';') +
1).trim().equals(s)) {
            cb[k].addItem(XmlDtd.matrix[k][j]);
          }
        }
        }
      }
    }

    public void actionPerformed(ActionEvent evt) {
      JComboBox cb = (JComboBox)evt.getSource();
      String str = (String)cb.getSelectedItem();
      cbrefresh(str);
    }

    public void mouseClicked(MouseEvent e) {
      String selnode = new String();
      int x = e.getX();
```

```
  int y = e.getY();
  boolean found = false;
  for (int i = 0 ; i < nnodes ; i++) {
      Node n = nodes[i];
      if (x == n.x & y == n.y) {
        if (n.type.indexOf('*') > 0) {
          selnode = n.lbl;
          found = true;
            }
      }
  }
  if (found) {
    OptionPanel(selind, selnode);
  }
}

public void mousePressed(MouseEvent e) {
    addMouseMotionListener(this);
  double bestdist = Double.MAX_VALUE;
  int x = e.getX();
  int y = e.getY();
  for (int i = 0 ; i < nnodes ; i++) {
      Node n = nodes[i];
      double dist = (n.x - x) * (n.x - x) + (n.y - y) * (n.y - y);
      if (dist < bestdist) {
        pick = n;
        bestdist = dist;
      }
  }
  pick.x = x;
  pick.y = y;
  repaint();
  e.consume();
}

public void mouseReleased(MouseEvent e) {
    removeMouseMotionListener(this);
    if (pick != null) {
        pick.x = e.getX();
        pick.y = e.getY();
        pick = null;
    }
  repaint();
  e.consume();
}

public void mouseEntered(MouseEvent e) {
}

public void mouseExited(MouseEvent e) {
```

```
    }

    public void mouseDragged(MouseEvent e) {
      pick.x = e.getX();
      pick.y = e.getY();
      repaint();
      e.consume();
    }

    public void mouseMoved(MouseEvent e) {
    }

    public void start() {
      mythread = new Thread(this);
      mythread.start();
    }

    public void stop() {
      mythread = null;
    }
}

public class XmlDtd extends Applet implements ActionListener {
    static MsgBox mess;
    XmlDtdPanel panel;
    ScrollPane sp;
    JPanel controlPanel;
    XmlDtdLegend legendPanel;
    JTextField fileName = new JTextField();
    JLabel label = new JLabel("Legend");
    JButton fileChooser_button = new JButton("Choose XML file");
    JButton mapping_button = new JButton("Mapping to DB2");
    private String[] retrievedFiles;
    private String retrievedDir = "\\dxx\\samples\\xml\\";
    private int kk1;
    private int kk2;
    private int kdtd1;
    static int icb;
    static int maxp;
    static String matrix[][] = new String[300][500]; // Combo Box items
    static String ncb[] = new String[50];      // line no from xml
    public void init() {
      setLayout(new BorderLayout());
      controlPanel = new JPanel();
      add("North", controlPanel);
      controlPanel.add(fileChooser_button);
      fileChooser_button.addActionListener(this);
      fileName.setColumns(40);
      fileName.setEnabled(false);
      controlPanel.add(fileName);
```

```
      controlPanel.add(mapping_button);
      mapping_button.addActionListener(this);
      legendPanel = new XmlDtdLegend(this);
      add("South",legendPanel);
      legendPanel.add(label);
      sp = new ScrollPane(ScrollPane.SCROLLBARS_AS_NEEDED);
      sp.setSize(size().width - 5Ø, size().height - 1);
      panel = new XmlDtdPanel(this);
      sp.add(panel);
      add("Center", sp);
      int len = 5Ø;
      String posx = "Ø1Ø1";
      String posy = "Ø1Ø1";
      String typefrom;
      String typeto;
      String edges = getParameter("edges");
      for (StringTokenizer t = new StringTokenizer(edges, ",") ;
t.hasMoreTokens() ; ) {
          String str = t.nextToken();
          int i = str.indexOf('-');
          if (i > Ø) {
            int j = str.indexOf('|');
            if (j > Ø) {
                posx = str.substring(j+1,j+5);
                posy = str.substring(j+5);
                str = str.substring(Ø, j);
            }
            typefrom = str.substring(Ø,1);
            if (str.substring(Ø,i).indexOf('*') > Ø) {
              typefrom = typefrom + '*';
            }
            typeto = str.substring(i+1,i+2);
            if (str.substring(i+1).indexOf('*') > Ø) {
              typeto = typeto + '*';
            }
            panel.addEdge(typefrom, str.substring(Ø,i), typeto,
str.substring(i+1), len, posx, posy);
          }
      }
    }

    public void destroy() {
        remove(panel);
        remove(controlPanel);
    }

    public void start() {
      panel.start();
    }
```

```
    public void stop() {
      panel.stop();
    }

    public void actionPerformed(ActionEvent e) {
      Object src = e.getSource();
      if (src == fileChooser_button) {
        JFileChooser chooser = new JFileChooser(retrievedDir);
        MyFileFilter filter = new MyFileFilter("xml", "XML Files");
        chooser.addChoosableFileFilter(filter);
        chooser.setDialogTitle("Select an XML file");
        int returnVal = chooser.showOpenDialog(this);
        if (returnVal == JFileChooser.APPROVE_OPTION) {
          String fileSelected = chooser.getCurrentDirectory() + "\\" +
chooser.getSelectedFile().getName();
          fileName.setText(fileSelected);
           try {
            String n[] = new String[300];
            String nleaf[] = new String[500];
            int i, j, k, il, minncb;
            k = -1;
            il = -1;
            minncb = -1;
            String fdtd;
            String hlp1 = new String();
            String hlp2 = new String();
            String hlp3 = new String();
            BufferedReader in1 = new BufferedReader(new
FileReader(fileSelected));
            String s = new String();
            boolean fnddoc = false;
            // Generate array of elements and attributes
            int idtd = -1;
            int jdtd = -1;
            int kdtd = -1;
            int inddtd = -1;
              String ndtd[] = new String[300];
              String ndtd1[] = new String[300];
            while ((s = in1.readLine()) != null) {
            if (s.indexOf("DOCTYPE") > 0) {
              if (s.indexOf(".dtd") > 0) {
                fnddoc = true;
                    i = s.indexOf('"');
                    j = s.indexOf(".dtd");
                fdtd = s.substring(i + 1, j + 4);
                    try {
                    BufferedReader indtd = new BufferedReader(new
FileReader(fdtd));
                    String sdtd = new String();
                       while ((sdtd = indtd.readLine()) != null &
```

```
(sdtd.indexOf("ELEMENT") < Ø) & (sdtd.indexOf("ATTLIST") < Ø)) {
                  }
                       do {
                    idtd++;
                    ndtd[idtd] = sdtd;
                    if (sdtd.indexOf('>') >= Ø) {
                    inddtd = idtd;
                    }
                  } while ((sdtd = indtd.readLine()) != null);
                    indtd.close();
                    } catch (FileNotFoundException edtd) {
                  mess = new MsgBox(new JFrame("") , "File Not Found:" +
fdtd);
                    } catch(IOException edtd) {
                  mess = new MsgBox(new JFrame("") , "IO Exception");
                    }
              } else {    // end if .dtd
                if (s.indexOf("[") >= Ø) {
                  fnddoc = true;
                    while ((s = in1.readLine()) != null) {
                    if (s.indexOf("]") >= Ø) break;
                    inddtd++;
                    ndtd[inddtd] = s;
                      }
                    }    // end if [
                  }
            }     // end if DOCTYPE
            if (fnddoc) break;
            }      // End while
            in1.close();
            if (!fnddoc) {
            mess = new MsgBox(new JFrame("") , "This kind of XML file
not supported !!!");
            return;
            }
            boolean fnddtd = false;
            String rootdtd = new String();
            for (jdtd = Ø ; jdtd <= inddtd; jdtd++) {
              fnddtd = false;
              if ((ndtd[jdtd].indexOf("<!ELEMENT") >= Ø) &
(ndtd[jdtd].indexOf('(') >= Ø) &
                (ndtd[jdtd].indexOf(')') >= Ø) &
(ndtd[jdtd].indexOf('#') < Ø)) {
              rootdtd =
ndtd[jdtd].substring(ndtd[jdtd].indexOf("<!ELEMENT") + 9,
ndtd[jdtd].indexOf('(')).trim();
                for (kdtd = Ø ; kdtd <= inddtd; kdtd++) {
                if (kdtd != jdtd) {
                  if ((ndtd[kdtd].indexOf("<!ELEMENT") >= Ø) &
(ndtd[kdtd].indexOf(rootdtd) >= Ø)) {
```

```
                    fnddtd = true;
                    break;
                    }
                  }
                }
                if (!fnddtd) break;
                }
              }
              if (!fnddtd) {
                kdtd1 = -1;
                  sortarray(1, rootdtd, inddtd, ndtd, ndtd1);
              }
                String s1 = new String();
              for (jdtd = 0 ; jdtd <= kdtd1; jdtd++) {
                   s1 = ndtd1[jdtd];
          if ((s1.indexOf("ELEMENT") > 0) | (s1.indexOf("ATTLIST") > 0)) {
                if (s1.indexOf("ELEMENT") > 0) {
                  i = s1.indexOf("ELEMENT");
                  j = s1.indexOf('(');
                  hlp2 = "e." + s1.substring(i + 7, j).trim();
                  for (int l = 0 ; l <= k ; l++) {
                        if (n[l].indexOf(hlp2 + '+') > 0) {
                    hlp2 = hlp2 + '+';
                    }
                        if (n[l].indexOf(hlp2 + '*') > 0) {
                      hlp2 = hlp2 + '*';
                    }
                  }
                  if (s1.indexOf("#PCDATA") <= 0) {
                    i = s1.indexOf(')');
                    hlp1 = s1.substring(j + 1, i);
                    hlp3 = hlp2;
                     for (StringTokenizer t = new StringTokenizer(hlp1,
","") ; t.hasMoreTokens() ; ) {
                      String str = "e." + t.nextToken().trim();
                      k = addarr(n, k, str, hlp2);
                     }
                   }
```

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

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

Engenio Information Technologies has announced Replication Express software for DB2. This new software is designed to improve back-up and recovery speed by automating the use of SANtricity Storage Management Software replication features such as Snapshot and Volume Copy.

Replication Express is policy-based software that provides users with an automated way to eliminate the back-up window.

For further information contact:
URL: www.engenio.com/default.aspx?pageID=713.

* * *

IBM has announced Version 8.3 of DB2 Content Manager. The new version has enhancements to its content management platform, and new workflow capabilities that help users automate processes using graphical tools.

Document routing integrates with workflow capabilities to help streamline business processes. The capture and management of XML documents in a common content repository can be automated.

The product integrates with Version 4.1.1 of DB2 Records Manager.

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

* * *

LiveTime Software has announced Version 3.5 of LiveTime Help Desk and LiveTime Support Desk. The new version includes support for WebSphere and DB2, as well as what it calls Quick Calls.

LiveTime now includes complete support for DB2 UDB V8.2 (and later) running on Solaris, Linux, and Windows platforms. The ability to support DB2 UDB allows an enterprise to standardize its Service Desk on the IBM technology platform. This release also includes enhanced support for IBM's WebSphere J2EE application server.

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

* * *

IBM has introduced software that enables a specific bundle of hardware, applications, and services to do policy-based automation of DB2 back-ups.

The initial version of the software requires a pSeries server, the AIX operating system, one or more instances of DB2 database files, a dedicated Tivoli Storage manager server and a DS4000 series storage array. Customers can buy a software key that allows them to set policies for automated back-up of the DB2 files.

For further information contact your local IBM representative.

* * *