DB2 Update

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

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

Subscriptions and back-issues
A year's subscription to DB2 Update, comprising twelve monthly issues, costs £255.00 in the UK; $380.00 in the USA and Canada; £261.00 in Europe; £267.00 in Australasia and Japan; and £265.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1997 issue, are available separately to subscribers for £22.50 ($33.50) 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/db2update.html; you will need to supply a word from the printed issue.

Editor
Trevor Eddolls

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

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

© Xephon plc 2001. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs $36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

Printed in England.
DB2 tablespace options

Although DB2 data is accessed at the table level, those skilled in DB2 database design and administration know that the actual data is stored in a structure known as a tablespace. Each tablespace correlates to one or more individual physical VSAM datasets that are used to house the actual DB2 data. When designing DB2 databases, DBAs can choose from three types of tablespaces, each one useful in different circumstances. The three types of tablespace are:

- Simple tablespaces
- Segmented tablespaces
- Partitioned tablespaces.

In general, the predominant tablespace type to use for most applications is the segmented tablespace. Segmented tablespaces provide a good combination of features that mix ease of use and set-up with performance and functionality. Many organizations adhere to standards stating that new DB2 tablespaces should be segmented tablespaces unless a compelling reason exists to choose one of the other tablespace types. You should consider using the other types of DB2 tablespaces in the following cases:

- Use partitioned tablespaces when you wish to encourage parallelism. Although DB2 can and will use parallel access techniques for non-partitioned tablespaces, partitioning data helps DB2 exploit parallelism.

- Consider using partitioned tablespaces when the amount of data to be stored is very large (more than 1 million pages). You will have more control over the placement of data in separate underlying datasets using partitioned tablespaces. This is often a concern with larger DB2 tables.

- Use partitioned tablespaces to reduce utility processing time and decrease contention. It is possible to execute DB2 utilities against single partitions without impacting concurrent access to data in other partitions. Furthermore, the utilities will run faster against
a single partition rather than against the entire tablespace and you will have more control over driving your utility workload. For example, you may not have sufficient time in the batch window to run a REORG on a four million page segmented tablespace, but you might have the time to run a REORG on one partition of that tablespace nightly. With four partitions of one million pages (or perhaps more partitions containing even fewer pages) you may be able to REORG one partition a night.

- Implement partitioned tablespaces to improve data availability. For example, if the data is partitioned by region, the partitions for the Eastern, Southern, and Northern regions can be made available while the Western region partition is being reorganized.

- Use partitioned tablespaces to improve recoverability. Once again, consider the ramifications if the data is partitioned by region. If an error impacts data for the Eastern region only, then only the Eastern partition needs to be recovered. The Southern, Northern, and Western regions can remain on-line, because they are not impacted by the problem in the Eastern region’s data.

- Consider partitioned tablespaces to isolate specific data areas in dedicated datasets. If there are specific data ‘hot spots’ that have higher data modification and/or access activity, you may be able to improve application performance by isolating the ‘hot spot’ into a single partition that can be tuned for the specific type of application access.

- Use a simple tablespace only when you need to mix data from different tables on one page. Simple tablespaces will mix data from each table assigned to the tablespace on each tablespace page. A segmented tablespace will not because each segment in the segmented tablespace is assigned to a single table. If you have two tables that are very frequently joined you might consider loading them into a single simple tablespace, ensuring that each row loaded from the first table is immediately followed by all of the rows from the second table that will be joined to the first table. This can minimize I/O for retrieval. However, DB2 will not maintain this ordering when the data is changed, so this approach is generally useful only for static data.
PARTITIONING CONSIDERATIONS

DB2 can handle up to 254 partitions per tablespace. The actual limit on the number of partitions depends on the DSSIZE of the tablespace. Large tablespaces are those that specify the LARGE parameter or have a DSSIZE greater than 4GB. The LARGE parameter was introduced with V5; DSSIZE with V6. A large tablespace can have from 1 to 254 partitions. Non-LARGE tablespaces are limited to no more than 64 partitions, as are any tablespaces created in a version prior to DB2 V5.

For non-LARGE partitioned tablespaces, the number of partitions impacts the maximum size of the dataset partition as follows:

<table>
<thead>
<tr>
<th>Number of partitions</th>
<th>Maximum dataset size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 16</td>
<td>4 GB</td>
</tr>
<tr>
<td>17 to 32</td>
<td>2 GB</td>
</tr>
<tr>
<td>33 to 64</td>
<td>1 GB</td>
</tr>
</tbody>
</table>

Keep these limitations in mind as you design your partitioned tablespaces.

As a general rule-of-thumb try to define tablespace partitions such that no one partition is more than 20 percent larger than the next largest partition. This provides even growth, which eases DASD monitoring and provides approximately even data access requirements and utility processing times across partitions. This is not a hard-and-fast rule though, especially when dealing with ‘hot spots’. The ‘hot spot’ partition may be much smaller than the other partitions, going against the idea of maintaining evenly distributed partitions. This is OK.

Deciding to use a partitioned tablespace is not as simple as merely determining the size of the table. In the early days of DB2, size was the primary consideration for choosing a partitioned tablespace. However, as DB2 has matured and the applications written using DB2 have been modernized, additional considerations will impact your partitioning decisions. Application-level details, such as data contention, performance requirements, degree of parallelism, and the volume of updates to columns in the partitioning index, must factor in the decision to use partitioned tablespaces.

Sometimes designers try to avoid partitioned tablespaces by dividing
a table into multiple tables, each with its own tablespace. This is not wise. Never attempt to avoid a partitioned tablespace by implementing several smaller tablespaces, each containing a subset of the total amount of data. When proceeding in this manner, the designer usually places separate tables into each of the smaller tablespaces. This almost always is a bad design decision because it introduces an uncontrolled and unneeded denormalization. Furthermore, when data that logically belongs in one table is separated into multiple tables, SQL operations to access the data as a logical whole are made needlessly complex. One example of this complexity is the difficulty in enforcing unique keys across multiple tables. Although partitioned tablespaces can introduce additional complexities into your environment, these complexities never outweigh those introduced by mimicking partitioning with several smaller, identical tablespaces. To clarify why this idea is bad, consider these two different ways of implementing a three ‘partition’ solution:

- The first, recommended, way is to create the table in a single partitioned tablespace with three partitions as follows:

  CREATE DB DB_SAMP;

  CREATE TABLESPACE TS_SAMP IN DB_SAMP
  ERASE NO NUMPARTS 3
  (PART 1
    USING STOGROUP SG_SAMP1
    PRIQTY 2000 SECQTY 50
    COMPRESS NO,

    PART 2
    USING STOGROUP SG_SAMP2
    PRIQTY 4000 SECQTY 150
    COMPRESS YES,

    PART 3
    USING STOGROUP SG_SAMP3
    PRIQTY 1000
    SECQTY 50
    COMPRESS YES)
  LOCKSIZE PAGE BUFFERPOOL BP1 CLOSE NO;

  CREATE TABLE TB_SAMP . . . IN DB_SAMP.TS_SAMP;

- The second, ill-advised, way is to create three tablespaces each with its own table as follows:
CREATE DB DB_SAMP;

CREATE TABLESPACE TS_SAMP1 IN DB_SAMP
    USING STOGROUP SG_SAMP1
    PRIQTY 2000  SECQTY 50
    ERASE NO COMPRESS NO
    LOCKSIZE PAGE   BUFFERPOOL BP1   CLOSE NO;

CREATE TABLESPACE TS_SAMP2 IN DB_SAMP
    USING STOGROUP SG_SAMP2
    PRIQTY 4000  SECQTY 150
    ERASE NO COMPRESS YES
    LOCKSIZE PAGE   BUFFERPOOL BP1   CLOSE NO;

CREATE TABLESPACE TS_SAMP3 IN DB_SAMP
    USING STOGROUP SG_SAMP3
    PRIQTY 1000  SECQTY 50
    ERASE NO COMPRESS YES
    LOCKSIZE PAGE   BUFFERPOOL BP1   CLOSE NO;

CREATE TABLE TB_SAMP1 . . . IN DB_SAMP.TS_SAMP1;
CREATE TABLE TB_SAMP2 . . . IN DB_SAMP.TS_SAMP2;
CREATE TABLE TB_SAMP3 . . . IN DB_SAMP.TS_SAMP3;

Now consider how difficult it would be to retrieve data in the second implementation if you did not know which ‘partition’ the data resides in, or if the data could reside in multiple partitions.

Using the first example a simple SELECT will work:

```
SELECT *
FROM TB_SAMP
WHERE COL1 = :HOST-VARIABLE;
```

In the second example, a UNION is required:

```
SELECT *
FROM TB_SAMP1
WHERE COL1 = :HOST-VARIABLE
UNION ALL
SELECT *
FROM TB_SAMP2
WHERE COL1 = :HOST-VARIABLE
UNION ALL
SELECT *
FROM TB_SAMP3
WHERE COL1 = :HOST-VARIABLE;
```

If other tables need to be joined the ‘solution’ becomes even more
complex. Likewise if data must be updated, inserted, or deleted and you do not know which ‘partition’ contains the impacted data. The bottom line is: avoid bypassing DB2 partitioning using your own pseudo-partitions.

PARTITIONING PROS AND CONS
Before deciding to partition a tablespace, weigh the pros and cons. Consult the following list of advantages and disadvantages before implementation:

Advantages of a partitioned tablespace:
- Each partition can be placed on a different DASD volume to increase access efficiency.
- Partitioned tablespaces are the only type of tablespace that can hold more than 64GB of data (the maximum size of simple and segmented tablespaces). A partitioned tablespace with extended addressability (EA-enabled) can hold up to 16 terabytes of data. Without being EA-enabled a partitioned tablespace can store up to about 1TB of data.
- Start and stop commands can be issued at the partition level. By stopping only specific partitions, the remaining partitions are available to be accessed thereby promoting higher availability.
- Free space (PCTFREE and FREEPAGE) can be specified at the partition level enabling the DBA to isolate data ‘hot spots’ to a specific partition and tune accordingly.
- Partitioning can optimize Query I/O, CPU, and Sysplex parallelism by removing disk contention as an issue because partitions can be spread out across multiple devices.
- Tablespace scans on partitioned tablespaces can skip partitions that are excluded, based on the query predicates. Skipping entire partitions can improve overall query performance for tablespace scans because less data needs to be accessed.
- The clustering index used for partitioning can be set up to decrease data contention. For example, if the tablespace will be
partitioned by DEPT, each department (or range of compatible departments) could be placed in separate partitions. Each department is in a discrete physical dataset, thereby reducing inter-departmental contention because of multiple departments coexisting on the same data page. Note that contention remains for data in non-partitioned indexes (although this contention has been significantly reduced in recent versions of DB2).

- DB2 creates a separate compression dictionary for each tablespace partition. Multiple dictionaries tend to cause better overall compression ratios. In addition, it is more likely that the partition-level compression dictionaries can be rebuilt more frequently than non-partitioned dictionaries. Frequent rebuilding of the compression dictionary can lead to a better overall compression ratio.

- The REORG, COPY, and RECOVER utilities can execute on tablespaces at the partition level. If these utilities are set to execute on partitions instead of on the entire tablespace, valuable time can be saved by processing only the partitions that need to be reorganized, copied, or recovered. Partition independence and resource serialization further increase the availability of partitions during utility processing.

Disadvantages of a partitioned tablespace:

- Only one table can be defined in a partitioned tablespace. This is not necessarily a disadvantage because most DBAs follow a one-table-per-tablespace rule.

- The columns of the partitioning index cannot be updated. To change a value in one of these columns, you must delete the row and then reinsert it with the new values.

- The range of key values for which data will be inserted into the table must be known and stable before you create the partitioning index. To define a partition, a range of values must be hard-coded into the partitioning index definition. These ranges will be used to distribute the data throughout the partitions. If you provide a stop-gap partition to catch all the values lower (or higher) than the defined range, monitor that partition to ensure that it does not
grow dramatically or cause performance problems if it is smaller or larger than most other partitions.

- After you define the method of partitioning, the only way to change it is to ALTER the partitioning index to change the LIMITKEY values and reorganize any impacted partitions. Prior to V6 you had to drop and redefine both the partitioning index and tablespace to change LIMITKEY specifications.

In general, partitioned tablespaces are becoming more useful. You might even want to consider using partitioning for most tablespaces (instead of segmented), especially if parallelism is an issue. At least, consider partitioning tablespaces that are accessed in a read-only manner by long-running batch programs. Of course, very small tablespaces are rarely viable candidates for partitioning, even with DB2’s advanced I/O, CPU, and Sysplex parallelism features. This is true because the smaller the amount of data to access, the more difficult it is to break it into pieces large enough such that concurrent, parallel processing will be helpful.

When using partitioned tablespaces, try to place each partition of the same partitioned tablespace on separate DASD volumes. Failure to do so can negatively affect the performance of query parallelism performed against those partitions. Disk drive head contention will occur because concurrent access is being performed on separate partitions that co-exist on the same device. Of course, with some of the newer storage devices, such as the ESS Shark hardware from IBM, dataset placement is a non-issue because of the way in which data is physically stored on the device.

SUMMARY

DB2 provides three different types of tablespaces, each of which has its own distinct set of advantages and disadvantages for use depending upon the situation. As a DBA you should understand the implementation details of each type of tablespace and be prepared to choose the right type of tablespace for each situation.

Craig S Mullins
Director, Technology Planning
BMC Software (USA) © Craig S Mullins 2001
Interpreting DB2 accounting trace using COBOL

There is no need to mention the wealth of accounting information present in the DB2 SMF accounting record (SMF 101). Many performance problems in an application can be quickly solved once the bottlenecks are identified. Performance monitoring tools help us identify bottlenecks, but what if a site does not have those expensive tools? What if a site wants to move to DB2 but does not want to budget for expensive tools right away. What if a site is planning to get rid of their performance tools to cut down costs? Sometimes it is difficult to justify the investment in performance monitoring tools because many sites using them do not use a lot of the information provided by the tools.

Here is a small utility program that would interpret the SMF 101 record (DB2 accounting record) and produce a report that could help analyse a performance problem or take proactive action. The best part is that the program is written in COBOL and so your program can easily be customized according to the requirements (populating a database for performance analysis, customizing reports etc).

SOFTWARE DEPENDENCIES

This program works well when run against the SMF 101 records from DB2 for OS/390 Version 6 and with the trace destination as SMF. However if your site has some other version of DB2, then you need to check the layouts of DSNDQWA0, DSNDQWHS, DSNDQWAC, DSNDQLAC, and DSNDQXST from the corresponding SDSNMACS library shipped with your version of DB2. If these tally with the layout used by the program (the program uses layouts from the SDSNMACS library shipped with Version 6), it can be used as is, otherwise you need to make modifications to the layout to make it work with your version of DB2.

The program takes SMF101 records as input. It is necessary to have the trace destination as SMF and not GTF because the program processes the SMF101 header information. If the trace destination is GTF then minor changes are required to the length and the header layout definitions.
GETTING STARTED

Before the program can produce any report, DB2 accounting trace (Classes 1, 2, and 3) should be activated with the destination set to SMF (this is the default destination for accounting trace). Additional constraints like PLAN, AUTHID can be applied to restrict the kind of data collected by the trace. The program also makes use of an additional variable, CUTOFF-CPU-TIME, which is currently set to 1.00. The value assigned to this variable is the time in seconds and is used to report only those threads/connections where the ‘in DB2 CPU time’ is greater than or equal to values assigned to CUTOFF-CPU-TIME.

This helps us to eliminate a lot of output that would otherwise get generated as a result of DB2 Command executions (say like -display thread, -start procedure, -stop procedure, etc) and like SQL statements that may not be of any interest from a monitoring point of view. You can set the value of this variable as per your requirements.

The program also calls an Assembler routine to request storage for the linkage section blocks. These few lines request a GETMAIN and pass the pointer to the calling program. This Assembler code is listed in stream with the JCL to compile and link in Appendix A.

If one does not want to use any Assembler code then an alternative way would be to have a COBOL program call our COBOL program so that the caller requests linkage section storage.

GENERATING ACCOUNTING REPORT

This program processes SMF 101 records. An SMF dataset containing SMF records, identified by DDname SMFDATA, is the input to the program. To have better control over the SMF records to be processed, the IBM-supplied utility IFASMFDP can be used. This utility can dump records of interest (SMF Type 101) corresponding to a particular date/time interval. Using this will also be useful if we want to collect SMF data for a few days to analyse, because less resource (both DASD and CPU) would be needed to generate accounting information because of the reduced volume of data to work with. See Appendix B for the required JCL.
The job produces two sets of output that can be used to analyse the accounting information. DD cards SYSOUT and BPACCT identify the output destinations.

The first set of reports identified by DD card SYSOUT gives us information about DB2 elapsed and CPU times, DB2 I/O done, I/O wait time within DB2, RDS accounting information like number of DML statements executed by each connection, and DDF accounting information.

The second set of information identified by DD card BPACCT tells us about Buffer Manager accounting. This gives us the breakdown of Get Pages issued by a connection on each of the buffer pools it uses. It also indicates synchronous read/write activity if any.

Sample reports are shown in Appendix C. Output is shown here as wrapped because of paper size limitations. The output generated by the program does not wrap over and so it can easily be taken to a spreadsheet for further analysis if required.

Fields on the report include:

- **Date** – date on which the accounting record was written (format YYDDD).
- **Time** – time at which the accounting record was written (format HH:MM:SS).
- **Jobname** – for batch this is job name, for TSO it is the log-on ID, and for CICS it is the user-id specified on the RCT.
- **Connect** – connection type (batch, TSO, utility, CICS).
- **Plan** – application plan name.
- **DB2ELAP** – elapsed time in DB2 (in seconds).
- **DB2CPU** – in DB2 CPU time (in seconds).
- **DB2IO** – total number of get pages requested by connection.
- **IO WAIT** – accumulated I/O elapsed wait time (in seconds).
• Lock Wait – accumulated wait time due to latch or local lock contention.

• #Selects – number of SELECT statements executed under the connection.

• #Inserts – number of INSERT statements executed under the connection.

• #Updates – number of UPDATE statements executed under the connection.

• #Deletes – number of DELETE statements executed under the connection.

• #Opens – number of OPEN cursor statements executed under the connection.

• #Fetches – number of FETCH requests executed under the connection.

• #Closes – number of CLOSE cursor statements executed under the connection.

• #Prepares – number of PREPARE statements executed under the connection.

• Remote Location – IP address of the remote location.

• SQL-S – SQLs sent from the server.

• SQL-R – SQLs received from the client.

• ROWS SENT – rows sent to the client.

• BYTES-S – number of bytes sent from the server to the client.

• BYTES-R – number of bytes received from the client.

• BPID – bufferpool identifier (BP0, BP1, etc).

• #GETP – number of get pages for the bufferpool.

• SYNC-Read – number of synchronous reads.

• SYNC-Write – number of synchronous writes.
The elapsed and CPU times shown in the report should not be used for chargebacks because the time interval is from thread connection to thread termination. It does not account for the time spent by an application before the connection or after the termination.

The program reports only three buffer pools, namely BP0, BP1, and BP2, but can be customized to report other/all buffer pools in a subsystem.

APPENDIX A

/YYYYY job card ****
//COMPILE EXEC PGM=ASMA9Ø,PARM='OBJECT,NODECK'
//SYSLIB DD DISP=SHR,DSN=SYS1.MACLIB
//SYSLIN DD DSN=&&LOADSET(GETSTORG),DISP=(NEW,PASS),
   // UNIT=SYSDA,SPACE=(800,(50,50,2)),DCB=(BLKSIZE=800)
//SYSPRINT DD SYSOUT=* //SYSUDUMP DD SYSOUT=* //SYSUT1 DD UNIT=SYSDA,SPACE=(800,(50,50),ROUND) //SYSUT2 DD UNIT=SYSDA,SPACE=(800,(50,50),ROUND) //SYSIN DD *
ASSEMBLER ROUTINE TO GET STORAGE
GETSTORG CSECT
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15
* PARMS: LENGTH OF STORAGE (4-BYTES BINARY - PIC S9(8) COMP)
* UPDATED ADDRESS (4-BYTES BINARY - USAGE POINTER )
STM R14,R12,12(R13) SAVE REGS
LR R12,R15 LOAD BASE REG
USING GETSTORG,R12 ADDRESSABILITY
LR R15,R13 OLD SAVE AREA ADDRESS
LA R13,SAVEAREA POINT TO NEW SAVE AREA
ST R15,4(R13) PUT OLD SAVE ADDR IN IT
L R2,0(R1) GET ADDRESS OF STORAGE LENGTH
ICM R2,15,0(R2) GET ACTUAL STORAGE LENGTH
L R3,4(R1) GET ADDRESS OF AREA TO UPDATE
GETMAIN EU, LV=(2), A=(3) GET STORAGE
L R13,4(R13) GET OLD SAVE AREA ADDRESS
LM R14, R12, R12(R13) RESTORE REGS
LA R15, 0 SET RC=0
BR R14 RETURN
SAVEAREA DC 18F'Ø' SAVE AREA

APPENDIX B

****** job card ****
IFASMFDP EXEC PGM=IFASMFDP
SYSPRINT DD SYSOUT=* SYSUDUMP DD SYSOUT=* DUMPIN DD DISP=SHR, DSN=INPUT.SMFDATA DUMPOUT DD DSN=OUTPUT.SMF1Ø1.DØ131, DISP=(NEW, CATLG), DCB=(LRECL=3276Ø, RECFM=VBS), SPACE=(CYL, (1, 5)) SYSSIN DD *, DCB=BLKSIZE=Ø INDD(DUMPIN, OPTIONS(DUMP)) OUTDD(DUMPOUT, TYPE(1Ø1)) DATE(2Ø01Ø31, 2Ø01Ø31) START(Ø8Ø0) END(18Ø0) /* SMF1Ø1 SMF1Ø1 EXEC PGM=SMF1Ø1 STEPLIB DD DSN=XXXXX.LOADLIB, DISP=SHR SYSTSPRT DD SYSOUT=* BPACCT DD SYSOUT=*, DCB=LRECL=85 SMFDATA DD DSN=OUTPUT.SMF1Ø1.Ø0131, DISP=SHR, DCB=(LRECL=3276Ø, RECFM=VBS) SYSPRINT DD SYSOUT=* SYSOUT DD SYSOUT=*, DCB=LRECL=22Ø SYSUDUMP DD SYSOUT=* 

APPENDIX C

MVS System:OSSF DB2 Sub-System:DB2P Time in Seconds
-----------------------------------------------------------------
<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Jobname</th>
<th>Connect</th>
<th>Plan</th>
<th>DB2 ELAP</th>
<th>DB2 CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ø1Ø31</td>
<td>19:24:35</td>
<td>$TSTDL11</td>
<td>UTILITY</td>
<td>DSNUTIL</td>
<td>548.73</td>
<td>40.72</td>
</tr>
<tr>
<td>Ø1Ø31</td>
<td>20:04:19</td>
<td>$TST5113</td>
<td>BATCH</td>
<td>PITAPLN1</td>
<td>1621.33</td>
<td>244.87</td>
</tr>
<tr>
<td>Ø1Ø31</td>
<td>20:05:16</td>
<td>$TST5113</td>
<td>UTILITY</td>
<td>DSNUTIL</td>
<td>32.30</td>
<td>11.60</td>
</tr>
<tr>
<td>Ø1Ø31</td>
<td>20:06:31</td>
<td>$TST5114</td>
<td>UTILITY</td>
<td>DSNUTIL</td>
<td>9.72</td>
<td>1.19</td>
</tr>
<tr>
<td>Ø1Ø31</td>
<td>20:22:53</td>
<td>SQRW.EXE</td>
<td>SERVER</td>
<td>DISTSERV</td>
<td>150.52</td>
<td>54.61</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB2IO</th>
<th>IO WAIT</th>
<th>Lock Wait #Selects #Inserts #Updates #Deletes</th>
</tr>
</thead>
<tbody>
<tr>
<td>36767</td>
<td>7.24</td>
<td>0.00</td>
</tr>
<tr>
<td>785845</td>
<td>51.23</td>
<td>0.05</td>
</tr>
<tr>
<td>27109</td>
<td>1.04</td>
<td>0.00</td>
</tr>
<tr>
<td>656</td>
<td>0.03</td>
<td>0.00</td>
</tr>
<tr>
<td>2707</td>
<td>0.82</td>
<td>0.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#Opens</th>
<th>#Fetches</th>
<th>#Closes</th>
<th>#Prepares</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>14435</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

Remote Location SQL-S SQL-R ROWS-S BYTES-S BYTES-R

172.10.12.14Ø 0 15 13 9368 9216
SMF101.COB

******************************************************************
***        PROGRAM FOR PROCESSING ACCOUNTING TRACE
******************************************************************
IDENTIFICATION DIVISION.
Program-Id.  SMF101.
Author.      Pranav Sampat.
*
ENVIRONMENT DIVISION.
*
INPUT-OUTPUT SECTION.
*
FILE-CONTROL.
  SELECT SMFDATA ASSIGN TO SMFDATA
  ORGANIZATION SEQUENTIAL.
  SELECT BPACCT ASSIGN TO BPACCT
  ORGANIZATION SEQUENTIAL.
*
DATA DIVISION.
FILE SECTION.
* SMF INPUT: QSAM, RECFM=VBS, BLKSIZE=4096 LRECL=32760
FD  SMFDATA
  LABEL RECORDS ARE STANDARD
  RECORDING MODE IS S
  BLOCK CONTAINS 4096 CHARACTERS
  RECORD CONTAINS 18 TO 32756 CHARACTERS.
Ø1  SMFDATA-MIN-RECORD    PIC X(18).
Ø1  SMFDATA-MAX-RECORD    PIC X(32756).
FD  BPACCT
  LABEL RECORDS ARE STANDARD.
Ø1  BPACCT-REC            PIC X(80).
WORKING-STORAGE SECTION.
*
Ø1  WS-BPACCT-REC.
  Ø5  F-STRDATE         PIC 99999.
  Ø5  FILLER            PIC X VALUE SPACE.
  Ø5  F-STRTIME.
    10  F-STRTIME-HH   PIC 99.
    10  FILLER         PIC X VALUE ':'.
    10  F-STRTIME-MM   PIC 99.
    10  FILLER         PIC X VALUE ':'.
    10  F-STRTIME-SS   PIC 99.
  Ø5  FILLER            PIC X VALUE SPACE.
  Ø5  F-JOBNAME         PIC X(8).
  Ø5  FILLER            PIC X VALUE SPACE.
  Ø5  F-CONNECT         PIC X(8).
  Ø5  FILLER            PIC X VALUE SPACE.
  Ø5  F-PLAN            PIC X(8).
  Ø5  FILLER            PIC X.
Ø5  BPID          PIC X(6).
Ø5  NUM-GETP       PIC ZZZZZZZ9.
Ø5  FILLER         PIC X.
Ø5  NUM-SYNC-READ  PIC ZZZZZZZ9.
Ø5  FILLER         PIC X.
Ø5  NUM-SYNC-WRITE  PIC ZZZZZZZ9.

*  
Ø1  WS-SMF-RECNO    PIC 9(6) COMP VALUE Ø.
Ø1  BIN4-32768      PIC 9(8) COMP VALUE 32768 SYNC.
Ø1  SMFØØ-PNTR     USAGE IS POINTER     SYNC.
Ø1  SMFØØ-ADDRVAL REDEFINES SMFØØ-PNTR     PIC S9(9) COMP.
Ø1  SMF1Ø1RPS-PNTR  USAGE IS POINTER     SYNC.
Ø1  SMF1Ø1RPS-ADDRVAL REDEFINES SMF1Ø1RPS-PNTR   PIC S9(9) COMP.
Ø1  DSNDQWHS-PNTR   USAGE IS POINTER     SYNC.
Ø1  DSNDQWHS-ADDRVAL REDEFINES DSNDQWHS-PNTR    PIC S9(9) COMP.
Ø1  DSNDQWAC-PNTR   USAGE IS POINTER     SYNC.
Ø1  DSNDQWAC-ADDRVAL REDEFINES DSNDQWAC-PNTR     PIC S9(9) COMP.
Ø1  DSNDQXST-PNTR   USAGE IS POINTER     SYNC.
Ø1  DSNDQXST-ADDRVAL REDEFINES DSNDQXST-PNTR    PIC S9(9) COMP.
Ø1  DSNDQUBAC-PNTR  USAGE IS POINTER     SYNC.
Ø1  DSNDQUBAC-ADDRVAL REDEFINES DSNDQUBAC-PNTR   PIC S9(9) COMP.
Ø1  DSNDQLAC-PNTR   USAGE IS POINTER     SYNC.
Ø1  DSNDQLAC-ADDRVAL REDEFINES DSNDQLAC-PNTR    PIC S9(9) COMP.
Ø1  END-OF-FILE-FLAG PIC X VALUE 'N'.
  88 END-OF-FILE VALUE 'Y'.
Ø1  BIN2-3Ø        PIC 9(4) COMP VALUE 1Ø1.
Ø1  FILLER REDEFINES BIN2-3Ø.
  Ø2  FILLER         PIC X(1).
  Ø2  BIN1-1Ø1       PIC X(1).
Ø1  WS-VARS.
  Ø2  PROGRAM-NAME   PIC X(8).
  Ø2  CUTOFF-CPU-TIME PIC 9V99 VALUE 1.ØØ.
  Ø2  TOT-INDB2-ELAP PIC 9(15)V99.
  Ø2  TOT-DB2-CPU    PIC 9(15)V99.
  Ø2  TOT-IOWT       PIC 9(15)V99.
  Ø2  TOT-LWT        PIC 9(15)V99.
  Ø2  TOT-IO         PIC 9(15)V99.
  Ø2  TEMP-HOURS     PIC 9(9) COMP.
  Ø2  TEMP-MIN       PIC 9(9) COMP.
  Ø2  TEMP-SECS      PIC 9(9) COMP.
  Ø2  WS-CNT         PIC 9(4) COMP.
  Ø2  WS-NUM-SELECTS PIC 9(9) COMP.
  Ø2  WS-NUM-INSERTS PIC 9(9) COMP.
  Ø2  WS-NUM-UPDATES PIC 9(9) COMP.
  Ø2  WS-NUM-DELETES PIC 9(9) COMP.
  Ø2  WS-NUM-OPENS   PIC 9(9) COMP.
  Ø2  WS-NUM-FETCHES PIC 9(9) COMP.
  Ø2  WS-NUM-CLOSES  PIC 9(9) COMP.
  Ø2  WS-NUM-PREPARES PIC 9(9) COMP.

Ø1  WS-HYPENS       PIC X(214) VALUE ALL '-'.

Ø1 OUT-REC-HEAD1.
  Ø5 FILLER PIC X(11) VALUE 'MVS System:'.
  Ø5 SYSTEM PIC X(4).
  Ø5 FILLER PIC X(5) VALUE SPACE.
  Ø5 FILLER PIC X(15) VALUE 'DB2 Sub-System:'.
  Ø5 SUBSYS PIC X(4).
  Ø5 FILLER PIC X(20) VALUE 'Time in Seconds'.Ø
Ø1 OUT-REC-HEADER.
  Ø5 FILLER PIC X(5) VALUE 'Date '.
  Ø5 FILLER PIC X(9) VALUE 'Time '.
  Ø5 FILLER PIC X VALUE SPACE.
  Ø5 FILLER PIC X(9) VALUE 'Jobname'.
  Ø5 FILLER PIC X(9) VALUE 'Connect'.
  Ø5 FILLER PIC X(10) VALUE 'Plan '.
  Ø5 FILLER PIC X(10) VALUE 'DB2 ELAP'.
  Ø5 FILLER PIC X(10) VALUE 'DB2 CPU'.
  Ø5 FILLER PIC X(9) VALUE 'DB2IO'.
  Ø5 FILLER PIC X(9) VALUE 'IO WAIT'.
  Ø5 FILLER PIC X(9) VALUE 'Lock Wait'.
  Ø5 FILLER PIC X(9) VALUE '#Selects'.
  Ø5 FILLER PIC X(9) VALUE '#Inserts'.
  Ø5 FILLER PIC X(9) VALUE '#Updates'.
  Ø5 FILLER PIC X(9) VALUE '#Deletes'.
  Ø5 FILLER PIC X(9) VALUE '#Opens'.
  Ø5 FILLER PIC X(9) VALUE '#Fetches'.
  Ø5 FILLER PIC X(10) VALUE '#Closes'.
  Ø5 FILLER PIC X(10) VALUE '#Prepares'.
  Ø5 FILLER PIC X(16) VALUE 'Remote Location'.
  Ø5 FILLER PIC X(7) VALUE 'SQL-S'.
  Ø5 FILLER PIC X(7) VALUE 'SQL-R'.
  Ø5 FILLER PIC X(8) VALUE 'ROWS-S'.
  Ø5 FILLER PIC X(8) VALUE 'BYTES-S'.
  Ø5 FILLER PIC X(8) VALUE 'BYTES-R'.
Ø1 BPACCT-REC-HEADER.
  Ø5 FILLER PIC X(8) VALUE 'Date '.
  Ø5 FILLER PIC X(8) VALUE 'Time '.
  Ø5 FILLER PIC X(9) VALUE 'Jobname'.
  Ø5 FILLER PIC X(9) VALUE 'Connect'.
  Ø5 FILLER PIC X(9) VALUE 'Plan '.
  Ø5 FILLER PIC X(8) VALUE 'BPID'.
  Ø5 FILLER PIC X(7) VALUE '#GETP'.
  Ø5 FILLER PIC X(10) VALUE 'SYNC-Read'.
  Ø5 FILLER PIC X(10) VALUE 'SYNC-Write'.
Ø1 OUT-REC-DATA.
  Ø5 STRDATE PIC 99999.
  Ø5 FILLER PIC X VALUE SPACE.
  Ø5 STRTIME.
    10 STRTIME-HH PIC 99.
    10 FILLER PIC X VALUE '. '.
    10 STRTIME-MM PIC 99.
    10 FILLER PIC X VALUE '. '.

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
* JOBNAME  JOB NAME,
* FOR BATCH JOB NAME,
* FOR TSO, QMF TSO LOGON ID,
* FOR CICS USER-ID SPECIFIED
* ON RCT
* CONNECT  CONNECTION TYPE
* FOR BATCH 'BATCH',
* FOR TSO 'TSO',
* FOR QMF 'DB2CALL'
* FOR UTILITY 'UTILITY'
* FOR CICS CICS-ID
* PLAN  APPLICATION PLAN NAM
*
* LINKAGE SECTION.
** GENERIC SMF RECORD.
 Ø1 SMF0Ø-RECORD.
 Ø2 SMF0ØFLG  PIC X(1).
 Ø2 SMF0ØRTY  PIC X(1).
 Ø2 SMF0ØTME  PIC 9(9) COMP.
 Ø2 SMF0ØDTE  PIC 9(7) COMP-3.
 Ø2 SMF0ØSID  PIC X(4).
 Ø2 SMF0ØSSI  PIC X(4).
 Ø2 SMF0ØSTP  PIC 9(4) COMP.
 Ø2 FILLER  PIC X(32743).
 Ø1 SMF1Ø1RPS.
** HEADER FOR TYPE 1Ø1 RECORD
 Ø2 SMF1Ø1RHD.
 Ø3 SMF1Ø1FLG  PIC X(1).
 Ø3 SMF1Ø1RTY  PIC X(1).
 Ø3 SMF1Ø1TME  PIC 9(9) COMP.
 Ø3 SMF1Ø1DTE  PIC 9(7) COMP-3.
 Ø3 SMF1Ø1SID  PIC X(4).
 Ø3 SMF1Ø1SSI  PIC X(4).
 Ø3 FILLER  PIC X(6).
** SELF-DEFINING SECTION OF TYPE 1Ø1 RECORD
 Ø2 DSNDQWAØ.
 Ø3 QWAØ1PSO  PIC 9(9) COMP.
 Ø3 QWAØ1PSL  PIC 9(4) COMP.
 Ø3 QWAØ1PSN  PIC 9(4) COMP.
 Ø3 QWAØ1R1O  PIC 9(9) COMP.
 Ø3 QWAØ1R1L  PIC 9(4) COMP.
 Ø3 QWAØ1R1N  PIC 9(4) COMP.
 Ø3 QWAØ1R2O  PIC 9(9) COMP.
 Ø3 QWAØ1R2L  PIC 9(4) COMP.
 Ø3 QWAØ1R2N  PIC 9(4) COMP.
 Ø3 QWAØ1R3O  PIC 9(9) COMP.
 Ø3 QWAØ1R3L  PIC 9(4) COMP.
 Ø3 QWAØ1R3N  PIC 9(4) COMP.
 Ø3 QWAØ1R4O  PIC 9(9) COMP.
 Ø3 QWAØ1R4L  PIC 9(4) COMP.
Ø3 QWA01R4N PIC 9(4) COMP.
Ø3 QWA01R5O PIC 9(9) COMP.
Ø3 QWA01R5L PIC 9(4) COMP.
Ø3 QWA01R5N PIC 9(4) COMP.
Ø3 QWA01R6O PIC 9(9) COMP.
Ø3 QWA01R6L PIC 9(4) COMP.
Ø3 QWA01R6N PIC 9(4) COMP.
Ø3 QWA01R7O PIC 9(9) COMP.
Ø3 QWA01R7L PIC 9(4) COMP.
Ø3 QWA01R7N PIC 9(4) COMP.
Ø3 QWA01R8O PIC 9(9) COMP.
Ø3 QWA01R8L PIC 9(4) COMP.
Ø3 QWA01R8N PIC 9(4) COMP.
Ø3 QWA01R9O PIC 9(9) COMP.
Ø3 QWA01R9L PIC 9(4) COMP.
Ø3 QWA01R9N PIC 9(4) COMP.

*****************************************************************
PRODUCT DATA SECTION
INSTRUMENTATION STANDARD HEADER
****************************************************************

Ø3 DSNDQWHS.
Ø2 QWHSLEN PIC 9(4) COMP.
Ø2 QWHSTYP PIC X(1).
Ø2 QWHSMRID PIC X(1).
Ø2 QWHSSIID PIC 9(4) COMP.
Ø2 QWHSNSDA PIC X(1).
Ø2 QWHSSRN PIC X(1).
Ø2 QWHSACE PIC 9(9) COMP.
Ø2 QWHSSSSID PIC X(4).
Ø2 QWHSSTCK.
   Ø3 QWHSSC1 PIC 9(9) COMP.
   Ø3 QWHSSC2 PIC 9(9) COMP.
Ø2 QWHSISEQ PIC 9(9) COMP.
Ø2 QWHSWSEQ PIC 9(9) COMP.
Ø2 QWHSMTN PIC X(4).
Ø2 QWHSLOCN PIC X(16).
Ø2 QWHSNID PIC X(8).
Ø2 QWHSLUNM PIC X(8).
Ø2 QWHSLUUV PIC X(6).
Ø2 QWHSLUCC PIC 9(4) COMP.

CORELATION HEADER
Ø2 QWHCLLEN PIC 9(4) COMP.
Ø2 QWHCTYP PIC X.
Ø2 FILLER PIC X.
Ø2 QWHCAID PIC X(8).
Ø2 QWHCCV PIC X(12).
Ø2 QWHCCN PIC X(8).
Ø2 QWHPLAN PIC X(8).
Ø2 QWHCPID PIC X(8).

*********** \n**** ACCOUNTING DATA \n***********

\n01 DSNDQWAC.
  \n  02 QWACBSC picture x(8).
  02 QWACESC picture x(8).
  02 QWACBJST picture 9(16) COMP.
  02 QWACEJST picture 9(16) COMP.
  02 QWACBSRB picture 9(16) COMP.
  02 QWACESRB picture 9(16) COMP.
  02 QWACRINV picture 9(9) COMP.
  02 QWACNID picture X(16).
  02 QWACCOMM picture 9(9) COMP.
  02 QWACABRT picture 9(9) COMP.
  02 QWCA1.
    \n    03 QWACASC picture 9(16) COMP.
    03 QWACAJST picture 9(16) COMP.
    03 QWACASRB picture 9(16) COMP.
    03 QWACAWTI picture 9(16) COMP.
    03 QWACAWTL picture 9(16) COMP.
    03 QWACARNA picture 9(9) COMP.
    03 QWACARNE picture 9(9) COMP.

**********
*** RDS STATISTICS BLOCK
**********

01 DSNDQXST.
  \n  02 OXHEAD.
    \n    03 QXID picture XX.
    03 QXLLEN picture 9(4) COMP.
    03 QXEYE picture X(4).
  02 OXSSTATS.
    \n    03 QXSELECT picture 9(9) COMP.
    03 QXINSRT picture 9(9) COMP.
    03 QXUPDTE picture 9(9) COMP.
    03 QXDELET picture 9(9) COMP.
    03 QXDESC picture 9(9) COMP.
    03 QXPREP picture 9(9) COMP.
    03 QXOPEN picture 9(9) COMP.
    03 QXCLOSE picture 9(9) COMP.
    03 QXCRTAB picture 9(9) COMP.
    03 QXCRINX picture 9(9) COMP.
    03 QXCRABS picture 9(9) COMP.
    03 QXCRSYN picture 9(9) COMP.
    03 QXCRDAB picture 9(9) COMP.
    03 QXCRSTG picture 9(9) COMP.
    03 QXDEFVU picture 9(9) COMP.
    03 QXDRPIX picture 9(9) COMP.
    03 QXDRPTA picture 9(9) COMP.
    03 QXDRPTS picture 9(9) COMP.
    03 QXDRPDAT picture 9(9) COMP.
*** BUFFER MANAGER ACCOUNTING BLOCK

***************

Ø1 DSNDQBAC.
Ø2 QBACPID PIC 9(9) COMP.
Ø2 QBACGET PIC 9(9) COMP.
Ø2 QBACBPX PIC 9(9) COMP.
Ø2 QBACSWS PIC 9(9) COMP.
Ø2 QBACSWU PIC 9(9) COMP.
Ø2 QBACRIO PIC 9(9) COMP.
Ø2 QBACSEQ PIC 9(9) COMP.
Ø2 QBACIMW PIC 9(9) COMP.
Ø2 QBACLPF PIC 9(9) COMP.
Ø2 QBACDPF PIC 9(9) COMP.

QBACPID BUFFER POOL ID
QBACGET # OF GET PAGE ISSUED
QBACBPX # OF BUFFER POOL EXPANSIONS required
QBACSWS # OF SETW ISSUED FOR SYSTEM pages
QBACSWU # OF SETW ISSUED FOR UW PAG
QBACRIO # OF SYNCHRONOUS READ I/O
QBACSEQ # OF SEQ PREFETCH REQUESTED
QBACIMW # OF SYNCHRONOUS WRITE I/O
QBACLPF # OF LIST PREFETCH REQUEST
QBACDPF # OF DYNAMIC PREFETCH REQUEST

***************

*** DISTRIBUTED DATA FACILITY ACCOUNTING BLOCK

***************

Ø1 DSNDQLAC.
Ø2 QLACLLOCN PIC X(16).
Ø2 QLACSQLS PIC 9(9) COMP.
Ø2 QLACSQLR PIC 9(9) COMP.
Ø2 QLACROWS PIC 9(9) COMP.
Ø2 QLACROWR PIC 9(9) COMP.
Ø2 QLACBYTES PIC 9(9) COMP.
Ø2 QLACBYTR PIC 9(9) COMP.
Ø2 QLACCNVS PIC 9(9) COMP.

QLACLLOCN - LOCATION
QLACSQLS - NUMBER OF SQL STATEMENTS SENT
QLACSQLR - NUMBER OF SQL STATEMENTS RECEIVED
QLACROWS - NUMBER OF ROWS SENT
QLACROWR - NUMBER OF ROWS RECEIVED
QLACBYTES - NUMBER OF BYTES SENT
* QLACBYTR - NUMBER OF BYTES RECEIVED
*
PROCEDURE DIVISION.
ØØØ-MAINLINE.
  PERFORM 1ØØ-INITIALIZE
  PERFORM UNTIL END-OF-FILE
    READ SMFDATA INTO SMFØØ-RECORD
    AT END MOVE 'Y' TO END-OF-FILE-FLAG
  END-READ
  IF NOT END-OF-FILE AND
    SMFØØRTY = BIN1-1Ø1
    ADD 1 TO WS-SMF-RECNO
    PERFORM 2ØØ-TYPE1Ø1-PROCESS
    PERFORM 6ØØ-DISPLAY-ACCOUNTING-INFO
  END-IF
  END-PERFORM.
CLOSE SMFDATA.
CLOSE BPACCT.
GOBACK.

1ØØ-INITIALIZE.
******************************************************************************
***** Request Storage for the linkage section                           *****
******************************************************************************
CALL 'GETSTORG' USING BIN4-32768 SMFØØ-PNTR.
SET ADDRESS OF SMFØØ-RECORD TO SMFØØ-PNTR.
OPEN INPUT SMFDATA.
OPEN OUTPUT BPACCT.
WRITE BPACCT-REC FROM BPACCT-REC-HEADER
WRITE BPACCT-REC FROM WS-HYPENS.

2ØØ-TYPE1Ø1-PROCESS.
******************************************************************************
** THE FOLLOWING CODE ADDRESSES THE HEADER AND SELF-DEFINING **
** SECTIONS**
******************************************************************************
MOVE SMFØØ-ADDRVAL TO SMF1Ø1RPS-ADDRVAL.
SET ADDRESS OF SMF1Ø1RPS TO SMF1Ø1RPS-PNTR.
*** 4 IS SUBTRACTED AS RDW IS NOT INCLUDED IN OUR RECORD
COMPUTE DSNDQWHS-ADDRVAL = SMF1Ø1RPS-ADDRVAL + QWAØ1PSO - 4
SET ADDRESS OF DSNDQWHS TO DSNDQWHS-PNTR.
** THE FOLLOWING CODE ADDRESSES THE ACCOUNTING DATA
MOVE SMF1Ø1SID TO SYSTEM
MOVE SMF1Ø1SSI TO SUBSYS
MOVE SMF1Ø1DTE TO STRDATE, F-STRDATE
PERFORM 2ØØ-CONVERT-TIME
MOVE TEMP-HOURS TO STRTIME-HH, F-STRTIME-HH
MOVE TEMP-MIN TO STRTIME-MM, F-STRTIME-MM
MOVE TEMP-SECS TO STRTIME-SS, F-STRTIME-SS
MOVE QWHCCN TO CONNECT, F-CONNECT
MOVE QWHCCV TO JOBNAME, F-JOBNAME

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
MOVE QWHCPLAN TO PLAN, F-PLAN
MOVE Ø TO TOT-INDB2-ELAP,
    TOT-DB2-CPU,
    TOT-IOWT,
    TOT-LWT
COMPUTE DSNDQWAC-ADDRVAL = SMF101RPS-ADDRVAL + QWA01R10 - 4
SET ADDRESS OF DSNDQWAC TO DSNDQWAC-PNTR
PERFORM 3000-COMPUTE-TIMES QWA01R1N TIMES
COMPUTE TOT-INDB2-ELAP =
    (TOT-INDB2-ELAP / 4096000000.0) + 0.005
COMPUTE TOT-DB2-CPU = (TOT-DB2-CPU / 4096000000.0) + 0.005
COMPUTE TOT-IOWT = (TOT-IOWT / 4096000000.0) + 0.005
COMPUTE TOT-LWT = (TOT-LWT / 4096000000.0) + 0.005
MOVE TOT-INDB2-ELAP TO DB2ELAP
MOVE TOT-DB2-CPU TO DB2CPU
MOVE TOT-IOWT TO DB2IOWAIT
MOVE TOT-LWT TO DB2LOCKWAIT
COMPUTE DSNDQXST-ADDRVAL = SMF101RPS-ADDRVAL + QWA01R20 - 4
SET ADDRESS OF DSNDQXST TO DSNDQXST-PNTR
MOVE Ø TO WS-NUM-SELECTS
    WS-NUM-INSERTS
    WS-NUM-UPDATES
    WS-NUM-DELETES
    WS-NUM-OPENS
    WS-NUM-FETCHES
    WS-NUM-CLOSES
    WS-NUM-PREPARES
PERFORM 3500-SQL QWA01R2N TIMES.
MOVE WS-NUM-SELECTS TO NUM-SELECTS
MOVE WS-NUM-INSERTS TO NUM-INSERTS
MOVE WS-NUM-UPDATES TO NUM-UPDATES
MOVE WS-NUM-DELETES TO NUM-DELETES
MOVE WS-NUM-OPENS TO NUM-OPENS
MOVE WS-NUM-FETCHES TO NUM-FETCHES
MOVE WS-NUM-CLOSES TO NUM-CLOSES
MOVE WS-NUM-PREPARES TO NUM-PREPARES
MOVE Ø TO TOT-IO.
COMPUTE DSNDQBAC-ADDRVAL = SMF101RPS-ADDRVAL + QWA01R30 - 4
SET ADDRESS OF DSNDQBAC TO DSNDQBAC-PNTR
PERFORM 4000-COMPUTE-IO QWA01R3N TIMES
MOVE TOT-IO TO DB2IO
MOVE ALL SPACES TO DDF-REC
MOVE '---------' TO REMOTE-LOCN
COMPUTE DSNDQLAC-ADDRVAL = SMF101RPS-ADDRVAL + QWA01R50 - 4
SET ADDRESS OF DSNDQLAC TO DSNDQLAC-PNTR
PERFORM 5000-DDF QWA01R5N TIMES.
2000-EXIT.
EXIT.

2100-CONVERT-TIME.
COMPUTE TEMP-HOURS = SMF101TME / 360000
COMPUTE TEMP-MIN =
(SMF101TME - (TEMP-HOURS * 360000)) / 6000
COMPUTE TEMP-SECS = (SMF101TME - (TEMP-HOURS * 360000) -
(TEMP-MIN * 6000)) / 100.

3000-COMPUTE-TIMES.
COMPUTE TOT-INDB2-ELAP = TOT-INDB2-ELAP + QWACASC
COMPUTE TOT-DB2-CPU = TOT-DB2-CPU + QWACAJST + QWACASRB
COMPUTE TOT-IOWT = TOT-IOWT + QWACAWTI
COMPUTE TOT-LWT = TOT-LWT + QWACAWTL
COMPUTE DSNDQWAC-ADDRVAL = DSNDQWAC-ADDRVAL + QWA01R1L
SET ADDRESS OF DSNDQWAC TO DSNDQWAC-PNTR.
3000-EXIT.
EXIT.

3500-SQL.
ADD QXSELECT TO WS-NUM-SELECTS
ADD QXINSRT TO WS-NUM-INSERTS
ADD QXUPDTE TO WS-NUM-UPDATES
ADD QXDELET TO WS-NUM-DELETES
ADD QXOPEN TO WS-NUM-OPENS
ADD QXFETCH TO WS-NUM-FETCHES
ADD QXCLOSE TO WS-NUM-CLOSES
ADD QXPREP TO WS-NUM-PREPARES
COMPUTE DSNDQXST-ADDRVAL = DSNDQXST-ADDRVAL + QWA01R3L
SET ADDRESS OF DSNDQXST TO DSNDQXST-PNTR.
3500-EXIT.
EXIT.

4000-COMPUTE-IO.
EVALUATE QBACPID
WHEN 0
  MOVE 'BP0' TO BPID
WHEN 1
  MOVE 'BP1' TO BPID
WHEN 2
  MOVE 'BP2' TO BPID
WHEN OTHER
  MOVE 'BPXX' TO BPID
END-EVALUATE
MOVE QBACGET TO NUM-GETP
MOVE QBACRIO TO NUM-SYNC-READ
MOVE QBACIMMW TO NUM-SYNC-WRITE
COMPUTE TOT-IO = TOT-IO + QBACGET + QBACSWS
COMPUTE DSNDQBAC-ADDRVAL = DSNDQBAC-ADDRVAL + QWA01R3L
SET ADDRESS OF DSNDQBAC TO DSNDQBAC-PNTR
IF TOT-DB2-CPU >= CUTOFF-CPU-TIME
  WRITE BPACCT-REC FROM WS-BPACCT-REC
END-IF.
4000-EXIT.
EXIT.
5000-DDF.
    MOVE QLACLOCN TO REMOTE-LOCN
    MOVE QLACSQLS TO SQLS-SENT
    MOVE QLACSQLR TO SQLS-RECEIVED
    MOVE QLACROWS TO ROWS-SENT
    MOVE QLACBYTS TO BYTES-SENT
    MOVE QLACBYTR TO BYTES-RECEIVED.
    COMPUTE DSNDQLAC-ADDRVAL = DSNDQLAC-ADDRVAL + QWA01R5L
    SET ADDRESS OF DSNDQLAC TO DSNDQLAC-PNTR.
5000-EXIT.
   EXIT.

6000-DISPLAY-ACCOUNTING-INFO.
   if ws-smf-recno = 1
       DISPLAY OUT-REC-HEAD1
       display ws-hypens
       DISPLAY OUT-REC-HEADER
       display ws-hypens
   end-if.
   ***************************************************************
   **** RECORD IS WRITTEN ONLY IF IN DB2 TIME IS MORE THAN 10 MS
   ***************************************************************
   if TOT-DB2-CPU >= CUTOFF-CPU-TIME
       DISPLAY OUT-REC-DATA
   end-if.

Pranav Sampat
Cognizant Technology Solutions (USA) © Xephon 2001

DB2 OLAP Server for OS/390 Version 7.1

IBM has made significant improvements to DB2 OLAP Server, as is shown by the increase in the Version number from 1.1 to 7.1! It is available for other platforms (see any IBM announcement for available hardware) but this article only discusses the mainframe version (OS/390 – old-timers will remember that mainframes were declared dead last decade).

OLAP is the acronym for On Line Analytical Processing. Ted Codd coined it in his 1993 paper Providing OLAP to User-Analysis: An IT Mandate although its foundation was established in the 1960s with IBM’s APL (A Programming Language). OLAP lets users model their business as a multidimensional cube whose typical dimensions are
geography, product, and time. OLAP comes in the following flavours:

- **DOLAP** – Desktop OLAP that usually employs two-tier architecture with data downloaded from a server for analysis on the client.

- **ROLAP** – Relational OLAP uses a relational database like DB2. Typical architecture is two-tier with a client generating a query and displaying the results.

- **MOLAP** – Multidimensional OLAP. Typical architecture does analysis on a mid-tier server like OS/390 using the Web for reporting. IBM defines DB2 OLAP Server as a MOLAP.

- **HOLAP** – Hybrid OLAP that attempts to combine the advantages of ROLAP and MOLAP. My view, contrary to IBM, is that DB2 OLAP Server is a HOLAP.

**WHY DB2 OLAP SERVER IS A HOLAP**

DB2 OLAP Server uses Essbase (Hyperion Solutions Corporation) as its multidimensional engine, but can store the data in DB2 allowing it to perform relational management functions such as back-up and recovery. Any OLAP application can also be assigned to DB2 using a star schema. (*A relational schema for a data warehouse, DB2 Update, Issue 86, December 1999* is my description of star schemas and variants.)

**STAR SCHEMA**

Star schema is divided into two table types of ‘fact’ and ‘dimension’. A fact table is at the star’s ‘centre’ whereas dimension tables are at its ‘points’. Fact tables contain columns to be measured such as sales and units. Dimension tables hold measuring columns such as date and location. A simple star table looks like:

```
date_dimkey location_dimkey $$ units
```

dimkey is a ‘foreign key’ to a dimension table such as date. DB2 OLAP Server usually creates four fact tables to hold all the data values for a relational cube.
Multidimensional cube software allows users to ask reporting questions (what happened when and where), do planning (what if we did this), and forecast (what next). Software also provides drill-down to details, drill-up to summaries or global views, pivot or slice-and-dice for different perspectives, and crosstabs for displaying data summaries based on their characteristics. DB2 enhanced SQL with new join and GROUP BY features can answer many multidimensional queries without needing multidimensional software. (See my articles All those joins, DB2 Update, Issue 97, November 2000, Have you been cubed?, DB2 Update, Issue 98, December 2000, and Have you been cubed? – further thoughts, DB2 Update, Issue 100, February 2001 for details and examples).

DB2 OLAP Server component Relational Storage Manager (RSM) creates and manages star schema objects (tables, views, indexes, and calculated data) from a metaoutline that you can create using the OLAP Metaoutline tool (a component of DB2 OLAP Integration Server Desktop). The DB2 OLAP Server calculation engine and 100+ functions simplify cube population and query execution, helping to optimize performance.

ANCHOR DIMENSION

DB2 OLAP Server uses an anchor dimension to define the relational cube fact table structure. It should be ‘dense’ having few nulls (‘sparse’ is many nulls). Minimizing nulls improves storage efficiency (the DB2 OLAP Server default is the densest dimension). An anchor dimension cannot be deleted or changed after data is loaded without clearing all that data, so be careful choosing it!

STORAGE MANAGERS

DB2 OLAP Server allows you to specify either a multidimensional storage manager or a relational storage manager. Multidimensional is the default but each OLAP application can specify either. Both multidimensional and relational store calculation and report scripts, data load rules, and database outline (defines all database elements) in a file system. Multidimensional stores data in the file system
whereas relational stores it in a DB2 relational database using a star schema.

DESIGNING AN OLAP APPLICATION
The key step is defining your database outline, which contains:

- Dimension and member definitions.
- Dense/sparse dimension tags and attributes.
- Anchor dimension attributes.
- Calculations.
- Shared members.
- Roll-up rules.

Implementation strategies, techniques, and instructions for the above are provided in the *OLAP Database Administrator’s Guide, Volumes I & II* (SC27-0788 and SC27-0789).

DB2 OLAP STARTER KIT
The DB2 OLAP Starter Kit is a subset of the DB2 OLAP Server provided free with DB2 UDB V7. It allows you to create a limited OLAP application, which IBM hopes will be so successful that you will acquire the full functionality of DB2 OLAP Server. OLAP applications developed with the Starter Kit can be uploaded to DB2 OLAP Server.

DB2 UDB DATA WAREHOUSE CENTER
DB2 OLAP Server Version 7.1 has been designed to integrate with UDB Data Warehouse Center to manage your OLAP environment. The combined functionality includes:

- Registering and accessing data sources.
- Defining data extraction and transformation steps.
- Populating data warehouse and OLAP applications.
• Managing, interchanging, and reporting metadata.
• Mapping multiple source data into star schema or multidimensional cubes.
• Loading, pre-calculating, and reporting on cubes.

DB2 OLAP Server can use any data warehouse star schema as a data source but cannot use it as its relational cube.

WHY
There is much hype about OLAP. A 1999 study by IDC (IDC Information Access Tools: 1999 Worldwide Markets and Trends) forecast a compound growth rate of 43.7% to 2003. This is evidence that many organizations are using or planning to use OLAP seriously. The applications are endless including:

• E-commerce.
• Customer Relationship Marketing (CRM).
• Enterprise Resource Planning (ERP).
• Competitor Analysis.
• Empowering Employees.
• Quality Engineering.

Information Catalog Manager can extract and publish meaningful metadata for users.

I recommend accepting IBM’s marketing ploy to use the Starter Kit to develop a test OLAP application. You have little to lose and much to gain!

CONCLUSION
Future articles will discuss various strategies for getting the most out of DB2 OLAP Server and DB2 UDB Data Warehouse Center.

Eric Garrigue Vesely  
Principal/Analyst  
Workbench Consulting (Malaysia)  
© Xephon 2001
Sample user-defined functions

This article describes user-defined functions that are provided with DB2. A user-defined function is an extension to the SQL language. A user-defined function is very similar to a stored procedure or a host language subprogram or function. However, a user-defined function is often the better choice for an SQL application because you can invoke a user-defined function in an SQL statement. You can use a user-defined function in an SQL statement in the same way as built-in functions.

The user-defined function runs in the stored procedure’s address space that is established in the Workload Manager (WLM) environment. More information about Workload Manager is in red book *DB2 UDB for OS/390 Version 6 Management Tools Package*, SG24-5759-00.

To prepare a user-defined function for execution, you must perform these steps:

- Create the user-defined function with the SQL command `CREATE FUNCTION`…
- Precompile the user-defined function program and bind the DBRM into a package. You need to do this only if your user-defined function contains SQL statements. You do not need to bind a plan for the user-defined function.
- Compile the user-defined function program with a compiler that supports Language Environment and link-edit the appropriate Language Environment components with the user-defined function. You must also link-edit the user-defined function with RRSAF. The language interface module for RRSAF, DSNRLI, is shipped with the linkage attributes AMODE(31) and RMODE(ANY).
- `GRANT EXECUTE` authority on the user-defined function or package.

The following is a list of my user-defined functions. I wrote all these function in PL/I:
• AGE – returns age in a user-specified format.
• COUNTER – a function that increments a variable in the scratchpad each time it is invoked.
• CUMUL – a function that generates a cumulative column.
• DATEUDF – returns the current date or a user-specified date in a user-specified format.
• REVERSE – returns the input string in reverse format.
• TIMEUDF – returns the current time or a user-specified time in a user-specified format.

**AGE**

```sql
AGE(expression)
```

The schema is SYSADM.

The AGE function returns age in two formats. The first one is in ‘YYMMDD’ format, and the second one is in ‘YY YEAR(S) MM MONTH(S) DD DAY(S)’ format. The formula used to calculate AGE is:

```
AGE = CURRENT DATE - DATE(expression)
```

The expression must be a date.

For example: the current date is ‘2000-10-26’.

```sql
SELECT AGE(DATE('1957-08-18')) AGE1,
       AGE(DATE('1957-08-18'),'T') AGE2
FROM SYSIBM.SYSDUMMY1
```

Result:

```
AGE1        AGE2
----------  ------------------------------
43Ø2Ø8      43 YEARS 2 MONTHS 8 DAYS
```

**COUNTER**

```sql
COUNTER(expression)
```

The schema is SYSADM.

The COUNTER function increments an integer variable in the scratchpad each time it is invoked. If an integer value is not specified, then the COUNTER function increments by 1.

For example:

```
SELECT COUNTER() "INCREMENT BY 1",
       COUNTER(5) "INCREMENT BY 5", NAME
FROM SYSIBM.SYSTABLES
WHERE DBNAME='DSNDB06'
  AND TSNAME='SYSOBJ'
```

Result:

```
<table>
<thead>
<tr>
<th>INCREMENT BY 1</th>
<th>INCREMENT BY 5</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>SYSCONSTDEP</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>SYSAUXRELS</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>SYSDATATYPES</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>SYSROUTINES</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>SYSPARMS</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>SYSROUTINEAUTH</td>
</tr>
<tr>
<td>7</td>
<td>35</td>
<td>SYSTRIGGERS</td>
</tr>
<tr>
<td>8</td>
<td>40</td>
<td>SYSSCHEMMAUTH</td>
</tr>
</tbody>
</table>
```

CUMUL

```
CUMUL(expression)
```

The schema is SYSADM.

The CUMUL function returns a cumulative value for each row. The argument must be smallint, integer, decimal, or float.

For example:

```
SELECT ID,     CUMUL(ID)     "CUMUL ID",
       SALARY, CUMUL(SALARY) "CUMUL SALARY"
FROM Q.STAFF
WHERE JOB='MGR'
```

Result:

```
<table>
<thead>
<tr>
<th>ID</th>
<th>CUMUL ID</th>
<th>SALARY</th>
<th>CUMUL SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>18357.50</td>
<td>18357.500</td>
</tr>
<tr>
<td>30</td>
<td>40</td>
<td>17506.75</td>
<td>35864.250</td>
</tr>
</tbody>
</table>
```
The schema is SYSADM.

The DATEUDF function returns the date in one of the following formats:

- **Weekday** returns the English name for the day of the week, in mixed case.
- **Month** returns the full English name of the month, in mixed case.
- **Normal** returns the date in the default format ‘YYYY-MM-DD’.
- **Usa** returns the date in the format ‘MM-DD-YYYY’.
- **European** returns the date in format ‘DD.MM.YYYY’.
- **Short** returns the date in format ‘DD Mon YYYY’.
- **Long** returns the date in format ‘DD Month YYYY’.
- **Julian** returns the date in format ‘YYYYDDD’.
- **Days** returns the number of days, including the current day, in the format ‘DDD’ (no leading zeros).
The argument must be a date.

For example: the current date is ‘2000-10-27’.

```
SELECT DATEUDF(DATE('1957-08-18'),'W') AS DAY,
       DATEUDF(DATE('1957-08-18'),'M') AS MONTH,
       DATEUDF(DATE('1957-08-18'),'S') AS "SHORT DAY",
       DATEUDF(DATE('1957-08-18'),'L') AS "LONG DAY",
       DATEUDF(DATE('1957-08-18'),'J') AS JULIAN,
       DATEUDF(DATE('1957-08-18'),'E') AS EUROPEAN,
       DATEUDF(DATE('1957-08-18'),'U') AS USA,
       DATEUDF(DATE('1957-08-18'),'D') AS DAYS,
       DATEUDF(DATE('1957-08-18'),'N') AS DEFAULT
FROM SYSIBM.SYSDUMMY1;
```

Result:

<table>
<thead>
<tr>
<th>DAY</th>
<th>MONTH</th>
<th>SHORT DAY</th>
<th>LONG DAY</th>
<th>JULIAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunday</td>
<td>August</td>
<td>18 Aug 1957</td>
<td>18 August 1957</td>
<td>1957230</td>
</tr>
</tbody>
</table>

and:

<table>
<thead>
<tr>
<th>EUROPEAN</th>
<th>USA</th>
<th>DAYS</th>
<th>DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.08.1957</td>
<td>08/18/1957</td>
<td>230</td>
<td>1957-08-18</td>
</tr>
</tbody>
</table>

**REVERSE**

```
?REVERSE(expresion)?
```

The schema is SYSADM.

The REVERSE function returns an input string in reverse format. The argument must be a character string.

For example:

```
SELECT REVERSE('UDF SAMPLE') AS REVERSE
FROM SYSIBM.SYSDUMMY1
```

Result:

```
REVERSE
------------
ELPMAS FDU
```
TIMEUDF

The schema is SYSADM.

The TIMEUDF function returns the time in one of the following formats:

- **Normal** returns the time in the default format ‘hh:mm:ss’.
- **Minutes** returns the number of minutes since midnight in the format ‘mmmm’ (no leading zeros).
- **Seconds** returns the number of seconds since midnight in the format ‘ssss’ (no leading zeros).
- **D** returns the time in the format ‘hhmmss’.
- **Civil** returns the time in the Civil format ‘hh:mm am/pm’.

For example:

```
SELECT TIMEUDF(TIME(CURRENT TIME),'N') AS DEFAULT,
       TIMEUDF(TIME(CURRENT TIME),'M') AS MINUTES,
       TIMEUDF(TIME(CURRENT TIME),'S') AS SECONDS,
       TIMEUDF(TIME(CURRENT TIME),'D') AS LOCAL,
       TIMEUDF(TIME(CURRENT TIME),'C') AS CIVIL
FROM SYSIBM.SYSDUMMY1 WITH UR
```

Result:

<table>
<thead>
<tr>
<th>DEFAULT</th>
<th>MINUTES</th>
<th>SECONDS</th>
<th>LOCAL</th>
<th>CIVIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>14:45:22</td>
<td>885</td>
<td>53122</td>
<td>144522</td>
<td>2:45pm</td>
</tr>
</tbody>
</table>

THE UDF (USER-DEFINED FUNCTION) OBJECTS

The first step is to define my sample UDFs to DB2.

```
CREATE FUNCTION SYSADM.AGE
    ( DATE )
RETURNS  VARCHAR(10)
```

CREATE FUNCTION SYSADM.AGE
( DATE , VARCHAR(1) )
RETURNS VARCHAR(30)
SPECIFIC AGED
EXTERNAL NAME 'AGED' -- MVS load module
LANGUAGE PLI
PARAMETER STYLE DB2SQL
DETERMINISTIC
READS SQL DATA
DBINFO
FENCED
COLLID AGED
WLM ENVIRONMENT DSNNWLM1 -- WLM application
STAY RESIDENT YES
PROGRAM TYPE MAIN
NO EXTERNAL ACTION
RETURNS NULL ON NULL INPUT
NO SCRATCHPAD
NO FINAL CALL
DISALLOW PARALLEL
ASUTIME NO LIMIT
SECURITY DB2;

CREATE FUNCTION SYSADM.COUNTER
RETURNS INTEGER
SPECIFIC COUNTER
EXTERNAL NAME 'COUNTER' -- MVS load module
CREATE FUNCTION SYSADM.COUNTER
  ( INTEGER )
RETURNS  INTEGER
SPECIFIC COUNTERV
EXTERNAL NAME 'COUNTERS'  -- MVS load module
LANGUAGE PLI
PARAMETER STYLE DB2SQL
NOT DETERMINISTIC
NO SQL
NO DBINFO
FENCED
NO COLLID
WLM ENVIRONMENT DSNWLM1  -- WLM application
STAY RESIDENT YES
PROGRAM TYPE MAIN
NO EXTERNAL ACTION
RETURNS NULL ON NULL INPUT
SCRATCHPAD 100
NO FINAL CALL
DISALLOW PARALLEL
ASUTIME NO LIMIT
SECURITY DB2;

CUMUL

CREATE FUNCTION SYSADM.CUMUL
  ( INTEGER )
RETURNS  INTEGER
SPECIFIC CUMULI
EXTERNAL NAME 'CUMULI'  -- MVS load module
LANGUAGE PLI
PARAMETER STYLE DB2SQL
NOT DETERMINISTIC
NO SQL
NO DBINFO
FENCED
NO COLLID
WLM ENVIRONMENT DSNWLM1 -- WLM application
STAY RESIDENT YES
PROGRAM TYPE MAIN
NO EXTERNAL ACTION
RETURNS NULL ON NULL INPUT
SCRATCHPAD 100
NO FINAL CALL
DISALLOW PARALLEL
ASUTIME NO LIMIT
SECURITY DB2;

CREATE FUNCTION SYSADM.CUMUL
( DECIMAL(15,3) )
RETURNS DECIMAL(15,3)
SPECIFIC CUMULD
EXTERNAL NAME 'CUMULD' -- MVS load module
LANGUAGE PLI
PARAMETER STYLE DB2SQL
NOT DETERMINISTIC
NO SQL
NO DBINFO
FENCED
NO COLLID
WLM ENVIRONMENT DSNWLM1 -- WLM application
STAY RESIDENT YES
PROGRAM TYPE MAIN
NO EXTERNAL ACTION
RETURNS NULL ON NULL INPUT
SCRATCHPAD 100
NO FINAL CALL
DISALLOW PARALLEL
ASUTIME NO LIMIT
SECURITY DB2;

CREATE FUNCTION SYSADM.CUMUL
( DOUBLE )
RETURNS DOUBLE
SPECIFIC CUMULF
EXTERNAL NAME 'CUMULF' -- MVS load module
LANGUAGE PLI
PARAMETER STYLE DB2SQL
NOT DETERMINISTIC
NO SQL
NO DBINFO
FENCED
CREATE FUNCTION SYSADM.DATEUDF
  ( DATE
   , VARCHAR(1) )
RETURNS  VARCHAR(20)
SPECIFIC DATEUDF
EXTERNAL NAME 'DATEUDF'  -- MVS load module
LANGUAGE PLI
PARAMETER STYLE DB2SQL
DETERMINISTIC
READS SQL DATA
DBINFO
FENCED
COLLID DATEUDF
WLM ENVIRONMENT DSNNWLM1  -- WLM application
STAY RESIDENT YES
PROGRAM TYPE MAIN
NO EXTERNAL ACTION
RETURNS NULL ON NULL INPUT
NO SCRATCHPAD
NO FINAL CALL
DISALLOW PARALLEL
ASUTIME NO LIMIT
SECURITY DB2;

CREATE FUNCTION SYSADM.DATEUDF
  ( DATE )
RETURNS  VARCHAR(10)
SPECIFIC DATEUDFD
EXTERNAL NAME 'DATEUDFD'  -- MVS load module
LANGUAGE PLI
PARAMETER STYLE DB2SQL
DETERMINISTIC
READS SQL DATA
DBINFO
FENCED
COLLID DATEUDFD
CREATE FUNCTION SYSADM.REVERSE
   ( VARCHAR(4046) )
RETURNS  VARCHAR(4046)
SPECIFIC REVERSE
EXTERNAL NAME 'REVERSE'  -- MVS load module
LANGUAGE PLI
PARAMETER STYLE DB2SQL
DETERMINISTIC
NO SQL
DBINFO
FENCED
NO COLLID

CREATE FUNCTION SYSADM.TIMEUDF
   ( TIME
     , VARCHAR(1) )
RETURNS  VARCHAR(10)
SPECIFIC TIMEUDF
EXTERNAL NAME 'TIMEUDF'  -- MVS load module
LANGUAGE PLI
PARAMETER STYLE DB2SQL
DETERMINISTIC
READS SQL DATA
DBINFO
CREATE FUNCTION SYSADM.TIMEUDF
   ( TIME )
RETURNS  VARCHAR(10)
SPECIFIC TIMEUDFD
EXTERNAL NAME 'TIMEUDFD'  -- MVS load module
LANGUAGE PLI
PARAMETER STYLE DB2SQL
DETERMINISTIC
READS SQL DATA
DBINFO
FENCED
COLLID TIMEUDFD
WLM ENVIRONMENT DSNNWLM1  -- WLM application
STAY RESIDENT YES
PROGRAM TYPE MAIN
NO EXTERNAL ACTION
RETURNS NULL ON NULL INPUT
NO SCRATCHPAD
NO FINAL CALL
DISALLOW PARALLEL
ASUTIME NO LIMIT
SECURITY DB2;

LOAD MODULES

AGE – PL/I source code

* PROCESS SYSTEM(MVS);
AGE: PROC(UDF_parm1, UDF_RESULT,
   UDF_IND1, UDF_INDR,
   UDF_SQLSTATE, UDF_NAME, UDF_SPEC_NAME,
   UDF_DIAG_MSG, UDF_SCRATCHPAD,
   UDF_CALL_TYPE, UDF_DBINFO)
   OPTIONS(MAIN NOEXECOPS REENTRANT);
   /*********************************************************************************/
/* UDF : AGE                                                 */
/* INPUT : UDF_PARM1    CHAR(10)                              */
/* OUTPUT: UDF_RESULT   VARCHAR(10)                           */
/*********************************************************************/
DCL UDF_PARM1      CHAR(10);          /* INPUT PARAMETER           */
DCL UDF_RESULT     CHAR(10) VAR;      /* RESULT PARAMETER          */
DCL UDF_IND1       BIN FIXED(15);     /* INDICATOR FOR INPUT PARM */
DCL UDF_INDR       BIN FIXED(15);     /* INDICATOR FOR RESULT     */
DCL 1 UDF_SCRATCHPAD,                 /* SCRATCHPAD                */
3 UDF_SPAD_LEN     BIN FIXED(31),     /* SCRATCHPAD                */
3 UDF_SPAD_TEXT    CHAR(100);        /* SCRATCHPAD                */
EXEC SQL INCLUDE UDFINFO;                  /* DBINFO */
EXEC SQL INCLUDE SQLCA;

EXEC SQL SELECT CHAR(INTEGER(CURRENT DATE - DATE(:UDF_PARM1))) INTO :UDF_RESULT FROM SYSIBM.SYSDUMMY1 WITH UR;
END AGE;

AGED – PL/I source code

* PROCESS SYSTEM(MVS);
AGET: PROC(UDF_PARM1, UDF_PARM2, UDF_RESULT,
           UDF_IND1, UDF_INDR,
           UDF_SQLSTATE, UDF_NAME, UDF_SPEC_NAME,
           UDF_DIAG_MSG, UDF_SCRATCHPAD,
           UDF_CALL_TYPE, UDF_DBINFO)
       OPTIONS(MAIN NOEXECOPS REENTRANT);
/*********************************************************************/
/* UDF : AGET                                                 */
/* INPUT : UDF_PARM1    CHAR(10)                              */
/* : UDF_PARM2    VARCHAR(1)                                 */
/* OUTPUT: UDF_RESULT   VARCHAR(30)                           */
/*********************************************************************/
DCL UDF_PARM1      CHAR(10);          /* INPUT PARAMETER           */
DCL UDF_PARM2      CHAR(1) VAR;       /* INPUT PARAMETER           */
DCL UDF_RESULT     CHAR(30) VAR;      /* RESULT PARAMETER          */
DCL UDF_IND1       BIN FIXED(15);     /* INDICATOR FOR INPUT PARM */
DCL UDF_INDR       BIN FIXED(15);     /* INDICATOR FOR RESULT     */
DCL 1 UDF_SCRATCHPAD,                 /* SCRATCHPAD                */
EXEC SQL INCLUDE UDFINFO; /* DBINFO */
DCL (ADDR,LENGTH,SUBSTR,NULL) BUILTIN;
EXEC SQL INCLUDE SQLCA;

/* *********************************************************** */
/* RETURNS AGE IN FORMAT 'YY YEAR(S) MM MONTH(S) DD DAY(S)' */
/* AGE((DATE('2000-08-18'),'T') -> 43 YEARS 2 MONTHS 2 DAYS */
/* *********************************************************** */

EXEC SQL SELECT
CASE
    WHEN(SMALLINT(YY)) > 1 THEN YY||' YEARS '
    ELSE YY||' YEAR '
END CONCAT
CASE
    WHEN(SMALLINT(MM)) > 1
    THEN STRIP(CHAR(SMALLINT(MM)))||' MONTHS '
    ELSE STRIP(CHAR(SMALLINT(MM)))||' MONTH '
END CONCAT
CASE
    WHEN(SMALLINT(DD)) > 1
    THEN STRIP(CHAR(SMALLINT(DD)))||' DAYS'
    ELSE STRIP(CHAR(SMALLINT(DD)))||' DAY '
END
INTO :UDF_RESULT
FROM(
    SELECT SUBSTR(AGE,1,LENGTH(AGE)-4) YY,
            SUBSTR(AGE,3,2) MM,
            SUBSTR(AGE,5,2) DD
    FROM (
        SELECT STRIP(CHAR(INTEGER(CURRENT DATE-DATE(:UDF_PARM1)))) AGE
        FROM SYSIBM.SYSDUMMY1 ) X ) Y;

END AGET;

Editor's note: the remaining UDFs will be published next month.

Bernard Zver (Slovenia) © Xephon 2001
Developer Solutions has announced new versions of HiT Software’s HiT OLEDB and ODBC DB2 middleware server products. Enhancements include monitoring of remote Windows server middleware activity including connection pooling.

The HiT Server Manager allows a remote developer or administrator to see OLE DB and ODBC DB2 SQL connections by DB2 server, calling application and User ID, start time of connections, status of connection, traffic volume over connection, and memory utilization. DB2 server connections can be pooled and reused to optimise performance. The HiT Server Manager can define the time-out and dropping of these pooled connections.

HiT OLEDB Server and HiT ODBC Server versions are available for DB2 UDB on its supported platforms; there is also an AS/400-specific version.

For further information contact:
Developer Solutions, Bradninch Court, Castle Street, Exeter EX4 3PL, UK.
Tel: (01392) 262626.
URL: http://www.developersolutions.co.uk.

---

Compuware has launched Version 7.0 of its XPEDITER/TSO automated testing tool for DB2 Stored Procedures. The debugging option intercepts S/390 DB2 stored procedures that originate from any type of application.

Testing is done in the actual DB2/Workload Manager address space, with no need for simulation of how the stored procedure will run. Developers can trap and step through Stored Procedures written in COBOL, Assembler, PL/I, and C.

Also, they can suspend and resume program execution at any point in the program, display and change program data variables, alter program logic, and set statement execution counts. The product works with XPEDITER/Code Coverage and File-AID DB2.

For further information contact:
Compuware, 31440 Northwestern Highway, Farmington Hills, MI 48334-2564, USA.
Tel: (248) 737 7300.

---

SteelEye has announced that LifeKeeper for Windows NT and Linux now works with DB2 Universal Database to provide reliability features for eServer xSeries Intel-based servers. The company worked with IBM to integrate the latest version of LifeKeeper Next Generation Enterprise Reliability platform for DB2 UDB running in Windows NT and Linux.

LifeKeeper monitors system, storage, and application health, maintains client connectivity, and provides continuous data access, regardless of where the clients reside. It’s designed to prevent system or application failures.

For further information contact:
SteelEye Technology, 2660 Marine Way, Suite 200, Mountain View, CA 94043, USA.
Tel: (877) 319 0108.