112 DB2

February 2002

In this issue

update

DB2 Update

Published by

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

North American office

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

Subscriptions and back-issues

A year's subscription to *DB2 Update*, comprising twelve monthly issues, costs £255.00 in the UK; $380.00 in the USA and Canada; £261.00 in Europe; £267.00 in Australasia and Japan; and £265.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1997 issue, are available separately to subscribers for £22.50 ($33.75) each including postage.

DB2 Update on-line

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

Editor

Trevor Eddolls

Disclaimer

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

Contributions

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

Printed in England.

DB2 710 utilities

INTRODUCTION

With DB2 710, IBM separated most of the utilities from the base DB2 product. Most utilities have been grouped in three separate independent products:

- Operational utilities – this product, program number 5655-E63 (FMID JDB771K), includes Copy, Load (including Cross Loader), Rebuild Index, Recover, Reorg Tablespace, Reorg Index, Runstats (enhanced with history), Stospace, and Unload (new).
- Diagnostic and Recovery utilities – this product, program number 5655-E62 (FMID JDB771M), includes Check Data, Check Index, Check LOB, Copy, CopyToCopy (new), Mergecopy, Modify Recovery, Modify Statistics (new), Rebuild Index, and Recover.
- Utilities Suite (FMID JDB771K + JDB771M), which includes all the utilities in the above two products.

All the utilities are shipped deactivated with the Base Engine. The corresponding product licences must be obtained to activate the specific utilities functions. However, all utilities are always available for execution on DB2 Catalog and Directory.

The following utilities, and all stand-alone utilities, are considered core utilities and are included and activated with DB2 V7: CATMAINT, DIAGNOSE, LISTDEF, OPTIONS, QUIESCE, REPAIR, REPORT, TEMPLATE, and DSNUTILS.

The enhancements to the utilities provide improved functionality and performance. This allows the utilities to compete with Independent Software Vendor (ISV) offerings.

The most significant changes are grouped around:

- Dynamic utility jobs using wildcarding and dynamic allocation – this provides better, more automated administration and usage of all the utilities.

- The new UNLOAD utility now includes many of the features, functions, and improved performance that users have been looking for.
- Enhancements to Online Reorg to minimize the period of time during which the data is not available.
- The new CopyToCopy utility which allows ‘offline’ image copies duplication.
- The ability to keep Statistics History, allowing better performance analysis.

DYNAMIC UTILITY JOBS

Development and maintenance of utility jobs can be very time-consuming and error-prone. Users primarily face three challenges:

- DD statements must be provided for all datasets.
- Within each DD statement, space, disposition, and other parameters must reflect the current situation.
- Utility statements must explicitly list all DB2 objects to be processed and these objects must be named accurately.

As new objects are constantly created, others are deleted, and since the sizes of most objects vary over time, it is particularly difficult to keep up with the changes.

DB2 710 addresses these three challenges by introducing the new utility control statements, TEMPLATE and LISTDEF.

With DB2 710, database administrators can submit utility jobs more quickly and easily.

Now you can:

- Dynamically allocate the datasets required to process those objects (TEMPLATE).
- Dynamically create object lists from a pattern-matching expression (LISTDEF).

The use of TEMPLATE utility control statements simplifies your JCL

by eliminating most dataset DD cards. Now you can provide dataset templates, and the DB2 product dynamically allocates the datasets that are required based on your allocation information.

Using a LISTDEF facility, you can standardize object lists and the utility control statements that refer to them. Standardization reduces the need to customize and change utility job streams over time.

Using these new functionalities, database administrators require less time to maintain utilities jobs.

TEMPLATE

With the TEMPLATE utility control statement, you define a dynamic list of dataset allocations.

More precisely:

- You create a skeleton or a pattern for the names of the datasets to allocate.
- The creation of the list of datasets to allocate is dynamic. That is, this list is generated each time the template is used by an executing utility. Therefore, a template automatically reflects the dataset allocations currently needed.
- The allocation of the datasets is dynamic. That is, the datasets are not allocated at the beginning of the job step, but each invoked utility allocates the datasets at execution time.
- You make use of a template by specifying the name of the template within a utility control statement – for example, during an image copy, within the COPY control statement.

In order to illustrate the use of the TEMPLATE statement, we will consider JCL to copy two partitions tablespaces of the DB2 catalog.

In DB2 Version 610, you should code something like:

```
//STEPØ1  EXEC DSNUPROC,PARM='DB2S,ICØØ'
//*
//SYSCOPØ1 DD DISP=(NEW,CATLG,DELETE),SPACE=(CYL,(1,1),RLSE),
//            DSN=SYSDB2.DB2S.DSNDBØ6.SYSGPAUT.ICØ1
//*
//SYSCOPØ2 DD DISP=(NEW,CATLG,DELETE),SPACE=(CYL,(1,1),RLSE),
```

```
//            DSN=SYSDB2.DB2S.DSNDBØ6.SYSGROUP.ICØ1
//*
//SYSIN    DD *
   COPY TABLESPACE DSNDBØ6.SYSGPAUT COPYDDN(SYSCOPØ1)
   COPY TABLESPACE DSNDBØ6.SYSGROUP COPYDDN(SYSCOPØ2)
/*
```

In DB2 Version 710, you could use TEMPLATE to generate dynamically primary image copy dataset names:

```
//STEPØ1  EXEC DSNUPROC,PARM='DB2S,ICØØ'
//*
//SYSIN    DD *
   TEMPLATE TIC DSN(SYSDB2.&SSID..&DB..&TS..D&JDATE..T&HOUR.&MINUTE.)

   COPY TABLESPACE DSNDBØ6.SYSGPAUT COPYDDN(TIC)
   COPY TABLESPACE DSNDBØ6.SYSGROUP COPYDDN(TIC)
/*
```

Then you get the following utility output:

```
1DSNUØØØI    DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = ICØØ
ØDSNUØ5ØI    DSNUGUTC -  TEMPLATE TIC
DSN(SYSDB2.&SSID..&DB..&TS..D&JDATE..T&HOUR.&SECOND.)
 DSNU1Ø35I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
ØDSNUØ5ØI    DSNUGUTC -  COPY TABLESPACE DSNDBØ6.SYSGPAUT COPYDDN(TIC)
 DSNU1Ø38I   DSNUGDYN - DATASET ALLOCATED.  TEMPLATE=TIC
                        DDNAME=SYSØØØØ1
                        DSN=SYSDB2.DB2S.DSNDBØ6.SYSGPAUT.D2ØØ1261.TØ8Ø3
 DSNU4ØØI    DSNUBBID - COPY PROCESSED FOR TABLESPACE DSNDBØ6.SYSGPAUT
                       NUMBER OF PAGES=3
                       AVERAGE PERCENT FREE SPACE PER PAGE = 13.ØØ
                       PERCENT OF CHANGED PAGES =  Ø.ØØ
                       ELAPSED TIME=ØØ:ØØ:ØØ
 DSNU428I    DSNUBBID - DB2 IMAGE COPY SUCCESSFUL FOR TABLESPACE
DSNDBØ6.SYSGPAUT
ØDSNUØ5ØI    DSNUGUTC -  COPY TABLESPACE DSNDBØ6.SYSGROUP COPYDDN(TIC)
 DSNU1Ø38I   DSNUGDYN - DATASET ALLOCATED.  TEMPLATE=TIC
                        DDNAME=SYSØØØØ2
                        DSN=SYSDB2.DB2S.DSNDBØ6.SYSGROUP.D2ØØ1261.TØ8Ø3
 DSNU4ØØI    DSNUBBID - COPY PROCESSED FOR TABLESPACE DSNDBØ6.SYSGROUP
                       NUMBER OF PAGES=3
                       AVERAGE PERCENT FREE SPACE PER PAGE = 15.33
                       PERCENT OF CHANGED PAGES =  Ø.ØØ
                       ELAPSED TIME=ØØ:ØØ:ØØ
 DSNU428I    DSNUBBID - DB2 IMAGE COPY SUCCESSFUL FOR TABLESPACE
DSNDBØ6.SYSGROUP
 DSNUØ1ØI    DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN
CODE=Ø
```

Dataset names

The first challenge we mentioned is to provide a correct dataset name for the DB2-generated DD statements.

This is done using substitution variables.

The TEMPLATE dataset names pattern can be specified using a broad set of substitution variables

Job-related variables are:

- JOBNAME
- STEPNAME
- USERID
- SSID – DB2 subsystem ID.

Utility-related variables are:

- UTIL – utility ID (truncated to 8 characters).
- ICTYPE – image copy type F or I.
- LOCREM – character L is used when the utility defines a COPYDDN ddname.

  The character R is used when the utility defines a RECOVERYDDN DDname.

  It is valid only for image copy templates.
- PRIBAK – character P is used when the utility defines a COPYDDN DDname.

  The character B is used when the utility defines a RECOVERYDDN DDname.

  It is valid only for image copy templates.

Object-related variables are:

- DB – database name
- TS – tablespace name
- IS – index space

- SN – space name
- LIST – name of the list LISTDEF
- SEQ – sequence number of the object in the list.

Date and time-related variables are:

- DATE – yyyyddd
- YEAR – yyyy
- MONTH – mm
- DAY – dd
- JDATE – yyyddd
- TIME – hhmmss
- HOUR – hh
- MINUTE – mm
- SECOND – ss.

You should notice that all date/time values are set using Greenwich Mean Time (GMT).

Space allocation

The second challenge we mentioned is the provision of the correct parameters for the DB2-generated dataset allocations. Most notably, the space parameters should reflect the current size of the tablespace or index space and the different space requirements for the different DASD datasets used by a particular utility.

As DB2 automatically allocates these datasets when using templates, it must also generate the correct space parameters for each dataset automatically, if you have not specified the space option in your template definition.

To this end, DB2 uses formulae which are specific for each utility and each dataset. The input values used in these formulae mostly come from the DB2 catalog tables. The high-used RBA is read at open time and it is maintained by the buffer manager. It is the most current value, updated before it is even written to the ICF catalog.

All these formulae are documented in the standard DB2 manuals.

LISTDEF

Having already replaced the DD statements and related disposition parameters by templates, we now address the next mentioned challenge when coding a utility job: how to generate a complete and accurate list of all DB2 objects to be processed by the utility. This can be done using the new LISTDEF utility.

With this utility control statement you define a dynamic list of DB2 objects, namely tablespaces, index spaces, or their partitions.

As mentioned before, the actual list is generated each time it is used by an executing utility. Therefore, a list automatically reflects which DB2 objects currently exist.

More precisely:

- You provide the rules or the algorithm used to generate the list of such DB2 objects.
- This list is dynamic. That is, the actual list is generated each time the list is used by an executing utility. Therefore, a list automatically reflects the existing DB2 objects.
- You make use of such a list by specifying the name of the list after the keyword LIST within a utility control statement – for example, within the COPY control statement.

In order to illustrate the use of the LISTDEF statement, we will consider the JCL to recover all tablespaces PTDB.PTTSRT*.

In DB2 Version 610, you should code something like:

```
//SYSIN DD *
   COPY TABLEPACE PTDB.PTTSRTB1
        TABLEPACE PTDB.PTTSRTB6
        TABLEPACE PTDB.PTTSRTS1
        TABLEPACE PTDB.PTTSRTS6
/*
```

In DB2 Version 710, you could use a LISTDEF statement to generate dynamically a tablespaces list:

```
//STEPØ1  EXEC DSNUPROC,PARM='DB2S,ICØØ'
```

```
//*
//SYSIN    DD *
   LISTDEF  LIC  INCLUDE TABLESPACE PTDB.PTTSRT%

   TEMPLATE TIC  DSN(SYSDB2.&SSID..&DB..&TS..D&JDATE..T&HOUR.&MINUTE.)

   COPY LIST LIC COPYDDN(TIC)
/*
```

Then, you get the following utility SYSOUT:

```
1DSNU000I    DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = IC00
0DSNU050I    DSNUGUTC -  LISTDEF LIC INCLUDE TABLESPACE PTDB.PTTSRT%
 DSNU1035I   DSNUILDR - LISTDEF STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  TEMPLATE TIC
DSN(SYSDB2.&SSID..&DB..&TS..D&JDATE..T&HOUR.&MINUTE.)
 DSNU1035I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    DSNUGUTC -  COPY LIST LIC COPYDDN(TIC)
 DSNU1038I   DSNUGDYN - DATASET ALLOCATED.  TEMPLATE=TIC
                         DDNAME=SYS00001
                         DSN=SYSDB2.DB2S.PTDB.PTTSRTB1.D2001264.T1349
 DSNU400I    DSNUBBID - COPY PROCESSED FOR TABLESPACE PTDB.PTTSRTB1
                        NUMBER OF PAGES=2
                        AVERAGE PERCENT FREE SPACE PER PAGE =  4.00
                        PERCENT OF CHANGED PAGES =  0.00
                        ELAPSED TIME=00:00:01
 DSNU1038I   DSNUGDYN - DATASET ALLOCATED.  TEMPLATE=TIC
                         DDNAME=SYS00002
                         DSN=SYSDB2.DB2S.PTDB.PTTSRTB6.D2001264.T1349
 DSNU400I    DSNUBBID - COPY PROCESSED FOR TABLESPACE PTDB.PTTSRTB6
                        NUMBER OF PAGES=2
                        AVERAGE PERCENT FREE SPACE PER PAGE =  4.00
                        PERCENT OF CHANGED PAGES =  2.38
                        ELAPSED TIME=00:00:00
 DSNU1038I   DSNUGDYN - DATASET ALLOCATED.  TEMPLATE=TIC
                         DDNAME=SYS00003
                         DSN=SYSDB2.DB2S.PTDB.PTTSRTS1.D2001264.T1349
 DSNU400I    DSNUBBID - COPY PROCESSED FOR TABLESPACE PTDB.PTTSRTS1
                        NUMBER OF PAGES=2
                        AVERAGE PERCENT FREE SPACE PER PAGE =  4.00
                        PERCENT OF CHANGED PAGES =  0.00
                        ELAPSED TIME=00:00:00
 DSNU1038I   DSNUGDYN - DATASET ALLOCATED.  TEMPLATE=TIC
                         DDNAME=SYS00004
                         DSN=SYSDB2.DB2S.PTDB.PTTSRTS6.D2001264.T1349
 DSNU400I    DSNUBBID - COPY PROCESSED FOR TABLESPACE PTDB.PTTSRTS6
                        NUMBER OF PAGES=2
                        AVERAGE PERCENT FREE SPACE PER PAGE =  4.00
                        PERCENT OF CHANGED PAGES =  2.38
                        ELAPSED TIME=00:00:00
 DSNU428I    DSNUBBID - DB2 IMAGE COPY SUCCESSFUL FOR TABLESPACE
```

```
PTDB.PTTSRTB1
 DSNU428I    DSNUBBID - DB2 IMAGE COPY SUCCESSFUL FOR TABLESPACE
PTDB.PTTSRTB6
 DSNU428I    DSNUBBID - DB2 IMAGE COPY SUCCESSFUL FOR TABLESPACE
PTDB.PTTSRTS1
 DSNU428I    DSNUBBID - DB2 IMAGE COPY SUCCESSFUL FOR TABLESPACE
PTDB.PTTSRTS6
 DSNU010I    DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN
CODE=0
```

You should note that the LISTDEF control statement does not support pattern-matching for either catalog or directory objects.

Combined TEMPLATE and LISTDEF

If you want a utility that processes a dynamic list of DB2 objects, that is a list which may increase or decrease over time, you can use LISTDEF and LIST.

As many online utilities require dataset allocations for each object they process, the number (and the parameters) of these allocations must be dynamic, too. That cannot be done with ordinary DD statements, but that is exactly what templates support: a dynamic number of dataset allocations, fitting to the object or to the list of objects processed by a utility.

In summary, the usage of LISTDEF/LIST often requires the usage of TEMPLATE:

```
//*
//* no DD statements
//*
//SYSIN DD *
LISTDEF COPYLIST INCLUDE TABLESPACE MYDB.TS* PARTLEVEL
TEMPLATE COPYTMPL DSN ( &DB..&TS..P&PART..&PRIBAC..D&JDATE. )
COPY LIST COPYLIST COPYDDN ( COPYTMPL,COPYTMPL )
/*
```

Storing LISTDEF and TEMPLATE statements in a dataset

Instead of using SYSIN to provide your LISTDEF and TEMPLATE statements, you can store these statements in library datasets. When using these datasets in your utility jobs, the default DD names are respectively SYSLISTD and SYSTEMPL:

```
//SYSLISTD DD=MYHLQ.LISTDEF(LD01),DISP=OLD
```

```
//SYSTEMPL DD=MYHLQ.TEMPLATE(TPØ1),DISP=OLD
//SYSIN DD *
QUIESCE LIST COPYLIST
COPY LIST COPYLIST COPYDDN(COPYTMPL,COPYTMPL)
/*
```

With MYHLQ.LISTDEF(LD01):

```
LISTDEF COPYLIST INCLUDE TABLESPACES PARTLEVEL TABLESPACE MYDB.TS*
```

And MYHLQ.TEMPLATE(TP01):

```
TEMPLATE COPYTMPL DSN(&DB..&TS..P&PART..&PB..D&JDATE..M&MINUTE.)
```

OPTIONS(PREVIEW)

In order to offer a complete and manageable solution when dealing with dynamic utility jobs, another utility control statement, OPTIONS, is provided to complement the functions offered by LISTDEF/LIST and TEMPLATE.

Specifying OPTIONS(PREVIEW) will expand the list definitions and the template definitions and report them to you.

Sample output:

```
1DSNUØØØI    DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = ICØØ
ØDSNUØ5ØI    DSNUGUTC -  OPTIONS PREVIEW
 DSNU1ØØØI   DSNUGUTC - PROCESSING CONTROL STATEMENTS IN PREVIEW MODE
 DSNU1Ø35I   DSNUILDR - OPTIONS STATEMENT PROCESSED SUCCESSFULLY
ØDSNUØ5ØI    DSNUGUTC -  TEMPLATE TIC
DSN(SYSDB2.&SSID..&DB..&TS..D&JDATE..T&HOUR.&MINUTE.)
 DSNU1Ø35I   DSNUJTDR - TEMPLATE STATEMENT PROCESSED SUCCESSFULLY
ØDSNUØ5ØI    DSNUGUTC -  COPY TABLESPACE DSNDBØ6.SYSGPAUT COPYDDN(TIC)
 DSNU1ØØ9I   DSNUGPVV - TEMPLATE TIC
DSN=SYSDB2.DB2S.DSNDBØ6.SYSGPAUT.D2ØØ1261.T124Ø
 DSNU1ØØ7I   DSNUGPVV - DATE/TIME VALUES MAY CHANGE BEFORE EXECUTION
ØDSNUØ5ØI    DSNUGUTC -  COPY TABLESPACE DSNDBØ6.SYSGROUP COPYDDN(TIC)
 DSNU1ØØ9I   DSNUGPVV - TEMPLATE TIC
DSN=SYSDB2.DB2S.DSNDBØ6.SYSGROUP.D2ØØ1261.T124Ø
 DSNU1ØØ7I   DSNUGPVV - DATE/TIME VALUES MAY CHANGE BEFORE EXECUTION
 DSNUØ1ØI    DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN
CODE=4
```

Conclusion

The benefits of these new utility control statements are evident: the development and maintenance of the jobs is easier; and as changes are

reflected automatically, they require less user activity and also reduce the possibility of errors.

NEW UNLOAD UTILITY

UNLOAD is a new utility that delivers better performance and more flexibility than DSNTIAUL.

It supports both tablespace and image copy as input and allows unloading of multiple partitions in parallel.

You can provide field selection for the unloaded data and the ordering and formatting options via SQL-like syntax.

UNLOAD also provides for record sampling and character encoding. By sampling, you can reduce the number of rows to be unloaded. This unloaded data can then be used, for example, to fill a test table. You can also specify a limit of rows to be unloaded, or both.

With the SHRLEVEL CHANGE option, unloading tablespaces does not affect your SQL applications, as used to be the case with REORG UNLOAD EXTERNAL. And unloading directly from copy datasets does not touch the data in the tablespace at all.

SHRLEVEL REFERENCE or CHANGE option

This option specifies the type of access to the tablespace or the partition allowed to other processes while the data is being unloaded.

For SHRLEVEL CHANGE, uncommitted rows, if they exist, will be unloaded.

For SHRLEVEL REFERENCE, the UNLOAD utility drains writers on the tablespace; when data is unloaded from multiple partitions, the drain lock will be obtained for all selected partitions in the UTILINIT phase.

Unloading certain rows only

Sampling

Within the FROM TABLE clause, you can specify the percentage of rows to be unloaded.

Limit rows to be unloaded

You can specify the maximum number of rows to be unloaded from a table using the LIMIT parameter.

UNLOAD JCL sample

```
//STEP01  EXEC DSNUPROC,PARM='DB2S,IC00'
//*
//SYSREC   DD DISP=(,CATLG,DELETE),DSN=SYSDB2.DB2S.UNLOAD.SYSREC,
//            SPACE=(TRK,(1,1))
//*
//SYSPUNCH DD DISP=(,CATLG,DELETE),DSN=SYSDB2.DB2S.UNLOAD.SYSPUNCH
//            SPACE=(TRK,(1,1))
//*
//SYSIN    DD *
 UNLOAD TABLESPACE DSNDB06.SYSDBAUT   SHRLEVEL CHANGE
        FROM TABLE SYSIBM.SYSDATABASE LIMIT 2
        WHEN (NAME LIKE 'DSNDB%')
/*
```

Then, you get the following result:

```
1DSNU000I  DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = IC00
0DSNU050I  DSNUGUTC - UNLOAD TABLESPACE DSNDB06.SYSDBAUT SHRLEVEL CHANGE
 DSNU650I  -DB2S DSNUUGMS -   FROM TABLE SYSIBM.SYSDATABASE LIMIT 2 WHEN
 DSNU650I  -DB2S DSNUUGMS -   (NAME LIKE 'DSNDB%')
 DSNU1202I -DB2S DSNUULXA - SAMPLING LIMIT HAS BEEN REACHED FOR TABLE
SYSIBM.SYSDATABASE
 DSNU253I   DSNUUNLD - UNLOAD PHASE STATISTICS - NUMBER OF RECORDS
UNLOADED=2 FOR TABLE SYSIBM.SYSDATABASE
 DSNU252I   DSNUUNLD - UNLOAD PHASE STATISTICS - NUMBER OF RECORDS
UNLOADED=2 FOR TABLESPACE DSNDB06.SYSDBAUT
 DSNU250I   DSNUUNLD - UNLOAD PHASE COMPLETE, ELAPSED TIME=00:00:00
 DSNU010I   DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

Unloading from copy datasets

The advantages when unloading from copy datasets are:

- You do not touch the user data in the tablespace, so you do not degrade the performance of the SQL applications.
- You can unload the data even if the tablespace is stopped.

The JCL to copy an image copy should look like:

```
//STEP01  EXEC DSNUPROC,PARM='DB2S,IC00'
//*
```

```
//SYSREC   DD DISP=(,CATLG,DELETE),DSN=SYSDB2.DB2S.UNLOAD.SYSREC,
//            SPACE=(TRK,(1,1))
//*
//SYSPUNCH DD DISP=(,CATLG,DELETE),DSN=SYSDB2.DB2S.UNLOAD.SYSPUNCH,
//            SPACE=(TRK,(1,1))
//*
//SYSIN    DD *
 UNLOAD TABLESPACE DSAMPLE.TEMP
        FROM TABLE TSAMPLE.EMP
        FROMCOPY SYSDB2.DB2S.DSAMPLE.TEMP.D2001264.T1351
/*
```

Then, you get the following SYSOUT:

```
1DSNU000I    DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = IC00
0DSNU050I    DSNUGUTC -  UNLOAD TABLESPACE DSAMPLE.TEMP
 DSNU650I  -DB2S DSNUUGMS -   FROM TABLE TSAMPLE.EMP
 DSNU650I  -DB2S DSNUUGMS -   FROMCOPY
SYSDB2.DB2S.DSAMPLE.TEMP.D2001264.T1351
 DSNU253I    DSNUUNLD - UNLOAD PHASE STATISTICS - NUMBER OF RECORDS
UNLOADED=8 FOR TABLE TSAMPLE.EMP
 DSNU252I    DSNUUNLD - UNLOAD PHASE STATISTICS - NUMBER OF RECORDS
UNLOADED=8 FOR TABLESPACE DSAMPLE.TEMP
 DSNU250I    DSNUUNLD - UNLOAD PHASE COMPLETE, ELAPSED TIME=00:00:00
 DSNU010I    DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN
CODE=0
```

ONLINE REORG ENHANCEMENTS

The major improvement is the new FASTSWITCH keyword.

This keyword for ONLINE REORG basically replaces the approximately three-second outage associated with the renaming of original and shadow dataset copies with a memory-speed switch of MVS catalog entries.

In the SWITCH phase, all datasets involved in the online REORG are no longer renamed, because this can be very time consuming. Instead, the catalog and the directory are updated to point to the shadow datasets rather than to the original datasets.

Prior to DB2 710

In the UTILINIT phase, *shadow objects* of the tablespace (or its partitions) and for the index spaces (or their partitions) are created. Strictly speaking, this is not true, because these shadow objects are not reflected in the catalog. What it means is that new *shadow*

datasets are created, one for each dataset of the original objects (or their partitions). The dataset names of these shadow datasets differ from the original dataset names insofar as their fifth qualifier, also referred to as *instance node*, is 'S0001' rather than 'I0001'. In the SWITCH phase, DB2 renames the original datasets and the shadow datasets. More specifically, the instance node of the original datasets, 'I0001', is renamed to a temporary one, 'T0001'; afterwards, the fifth qualifier of the shadow dataset, 'S0001', is renamed to 'I0001'.

During these renames, which take about two seconds each, SQL applications cannot access the tablespace. After the SWITCH phase, the applications can resume their processing on the new 'I0001' datasets.

In the UTILTERM phase, the datasets with 'T0001' are deleted because they are not needed any more.

DB2 710 : Fast SWITCH

A revised alternative to the processing is performed to speed up the SWITCH phase, thus making the phase less open to data access by others:

- Invocation of AMS to rename the datasets is eliminated.
- An optional process, the Fast SWITCH, takes place:
  - In the UTILINIT phase, DB2 creates shadow datasets. The fifth qualifier of these datasets is now 'J0001'.
  - In the SWITCH phase, DB2 updates the catalog and the object descriptor (OBD) from 'I' to 'J' to indicate that the shadow object has become the active or valid database object. During that time, SQL applications cannot access the tablespace.
  - After the SWITCH phase, the SQL applications can resume their processing, now on the new 'J0001' datasets.
  - In the UTILTERM phase, DB2 deletes the obsolete original datasets with the instance node 'I0001'.

COPYTOCOPY UTILITY

DB2 710 introduces a new utility, COPYTOCOPY. It provides you with the opportunity to make additional full or incremental image copies, duly recorded in SYSIBM.SYSCOPY, from a full or incremental image copy that was taken by the COPY utility.

This applies to tablespaces or indexes and includes inline copies made by the REORG or LOAD utility.

Starting with either the local primary or recovery site primary copy, COPYTOCOPY can make up to three copies of one or more of the following types of copy:

- Local primary
- Local back-up
- Recovery site primary
- Recovery site back-up.

The copies are used by the RECOVER utility when recovering a tablespace or index space to the most recent time or to a previous time. These copies can also be used by MERGECOPY, UNLOAD, and possibly a subsequent COPYTOCOPY execution.

An example of the COPYTOCOPY statement is:

```
COPYTOCOPY TABLESPACE MYDB.TS1 FROMLASTFULLCOPY RECOVERYDDN(mydd)
```

STATISTICS HISTORY

DB2 710 offers better elapsed time reporting and statistics for performance evaluations over time. It lets you keep a history of statistics, allowing for better proactive performance analysis capabilities and better object monitoring.

With these improvements, you can monitor growth over time and (with the help of additional information) determine whether objects need to change.

Statistics history is supported via nine new catalog tables. The RUNSTATS, REORG, LOAD, and REBUILD utilities use the new

keyword HISTORY to update the statistics history. And you can delete old statistics from the catalog history tables using the new MODIFY STATISTICS utility.

The new catalog tables that support historical statistics include:

- SYSIBM.SYSCOLDIST_HIST
- SYSIBM.SYSCOLUMNS_HIST
- SYSIBM.SYSINDEXPART_HIST
- SYSIBM.SYSINDEXES_HIST
- SYSIBM.SYSINDEXSTATS_HIST
- SYSIBM.SYSLOBSTATS_HIST
- SYSIBM.SYSTABLEPART_HIST
- SYSIBM.SYSTABLES_HIST
- SYSIBM.SYSTABSTATS_HIST

The new RUNSTAT syntax looks like:

```
//*
//STEPØ1  EXEC DSNUPROC,PARM='DB2S,ICØØ'
//*
//SYSIN    DD *
   LISTDEF  LRS  INCLUDE TABLESPACE DSNDBØ6.SYSGPAUT
                 INCLUDE TABLESPACE DSNDBØ6.SYSGROUP

   RUNSTATS TABLESPACE LIST LRS   HISTORY(ALL)

   MODIFY STATISTICS   LIST LRS   DELETE ALL AGE(1)
/*
```

This job produces the following SYSOUT:

```
1DSNUØØØI    DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = ICØØ
ØDSNUØ5ØI    DSNUGUTC - LISTDEF LRS INCLUDE TABLESPACE DSNDBØ6.SYSGPAUT
INCLUDE TABLESPACE DSNDBØ6.SYSGROUP
 DSNU1Ø35I   DSNUILDR - LISTDEF STATEMENT PROCESSED SUCCESSFULLY
ØDSNUØ5ØI    DSNUGUTC - RUNSTATS TABLESPACE LIST LRS HISTORY(ALL)
 DSNU1Ø33I   DSNUGULM - PROCESSING LIST ITEM: TABLESPACE
DSNDBØ6.SYSGPAUT
 DSNU61ØI  -DB2S DSNUSUTP -   SYSTABLEPART CATALOG UPDATE FOR
DSNDBØ6.SYSGPAUT SUCCESSFUL
 DSNU61ØI  -DB2S DSNUSUTB -   SYSTABLES CATALOG UPDATE FOR
SYSIBM.SYSRESAUTH SUCCESSFUL
```

```
 DSNU610I  -DB2S DSNUSUTS -   SYSTABLESPACE CATALOG UPDATE FOR
DSNDB06.SYSGPAUT SUCCESSFUL
 DSNU620I  -DB2S DSNUSDRA - RUNSTATS CATALOG TIMESTAMP = 2001-09-20-
10.51.49.183308
 DSNU1033I   DSNUGULM - PROCESSING LIST ITEM: TABLESPACE
DSNDB06.SYSGROUP
 DSNU610I  -DB2S DSNUSUTP -   SYSTABLEPART CATALOG UPDATE FOR
DSNDB06.SYSGROUP SUCCESSFUL
 DSNU610I  -DB2S DSNUSUTB -     SYSTABLES CATALOG UPDATE FOR
SYSIBM.SYSSTOGROUP SUCCESSFUL
 DSNU610I  -DB2S DSNUSUTB -     SYSTABLES CATALOG UPDATE FOR
SYSIBM.SYSVOLUMES SUCCESSFUL
 DSNU610I  -DB2S DSNUSUTS -   SYSTABLESPACE CATALOG UPDATE FOR
DSNDB06.SYSGROUP SUCCESSFUL
 DSNU620I  -DB2S DSNUSDRA - RUNSTATS CATALOG TIMESTAMP = 2001-09-20-
10.51.59.387729
0DSNU050I    DSNUGUTC -  MODIFY STATISTICS LIST LRS DELETE ALL AGE(1)
 DSNU1033I   DSNUGULM - PROCESSING LIST ITEM: TABLESPACE
DSNDB06.SYSGPAUT
 DSNU1307I -DB2S DSNUMSTA - 4 SYSIBM.SYSTABLEPART_HIST ROWS WERE DELETED
 DSNU1308I -DB2S DSNUMSTA - 4 SYSIBM.SYSTABLES_HIST ROWS WERE DELETED
 DSNU1300I -DB2S DSNUMSTA - MODIFY STATISTICS HAS COMPLETED SUCCESSFULLY
 DSNU1033I   DSNUGULM - PROCESSING LIST ITEM: TABLESPACE
DSNDB06.SYSGROUP
 DSNU1307I -DB2S DSNUMSTA - 4 SYSIBM.SYSTABLEPART_HIST ROWS WERE DELETED
 DSNU1308I -DB2S DSNUMSTA - 8 SYSIBM.SYSTABLES_HIST ROWS WERE DELETED
 DSNU1300I -DB2S DSNUMSTA - MODIFY STATISTICS HAS COMPLETED SUCCESSFULLY
 DSNU010I    DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN
CODE=0
```

Systems Programmer
(France)

Data tool for database management

INTRODUCTION

I have created an integrated tool called ‘DB2 Data Toolkit’. It carries out a series of functions and uses some enhancements introduced with DB2 V5. The tool can be used in several ways, depending on your needs.

The procedure can:

- Replicate data between two or more databases.

- Transfer data between DB2 located in different sites, by transferring a sequential dataset (dump ADRDSSU) that is a package containing all you need to rebuild the data environment.
- Clone the database data and bind statement.
- Download and archive the database and supply all you need for the restore environment.
- Back up data and archive on tape in an automatic procedure. The data can be restored in any DB2 with at least DB2 V5.

PROCEDURE DESCRIPTION

The functions described are realized by a series of batch procedures that are described below:

- Unload data with reorg unload external function. This process is realized with the Reorg utility in order to ensure good performance during the download.
- Update target creator in SYSPUNCH with a new load creator name.
- Copy all SYSPUNCH cards in a sequential dataset for a fast editing change.
- Overwrite the SYSPUNCH sequential dataset.
- Load data. During this phase you can choose whether or not to reuse your VSAM DB2 dataset in order to save time.
- Start tablespace in read/write mode.
- Build all bind statements (plan/package) from DB2 catalog.
- Back-up output from the unload job, in order to have a trace of unloaded data.
- Create a report of the selected objects.
- Dump all we have created (data, job, bind statements, reports, etc) with ADRSSU. The sequential file created can be retained or exported into another DB2 environment in a different subarea.

- Delete all work areas.

PARAMETER DESCRIPTION

The following parameters describe how you can customize the REXX EXEC batch procedure:

- Subsys – DB2 subsystem name.
- Dbname – database name.
- Creunl – unload table creator.
- Tsubsys – target DB2 subsystem.
- TDBname – target database name.
- Creloa – load table creator.
- ReusVS – reuse VSAM dataset.
- Ownerb – bind owner.
- Full – type of unload.
- Maxdd – max number of DD for job.
- KeepTime – time to keep back-up.
- Autosub Submit – unload/dump/delete.
- Jobname – jobname.
- Catnam DB2 – catalog creator.

CHECKLIST FOR INSTALLATION

Follow these steps to install the components of the REXX procedure:

- Allocate a USER.LIBRARY
- Copy all REXX, macro, PROC, and JCL into the USER.LIBRARY:
  - REXX: $DB2UN00, $DB2UN01, $DB2UN02, $DB2UN03, $DB2UN04, $DB2UN05, $DB2UN06, $DB2PAR0, $DB2ALL0, $DB2TBNA, $DB2RC00, $DB2FOR0, $DB2OUBK, $DB2BIND, $DB2SDSF.

  - Macro: $MDB2018, $MDB2021, $MDB2023, $MDB2024, $MDB2025, $MDB2026, $MDB2027, $MDB2031, $MDB2039, $MDB2040.
  - Proc: DB2REXX1.
  - Sample job: $DB2UN00.

- Customize $DB2PAR0 REXX for the global environment, setting highlighted variables.
- Customize proc DB2REXX1 according your environment.
- Customize job $DB2UN00 according your environment.
- Create a member IEBDDIN in the user.library and insert the string:

```
COPY OUTDD=OUT1,INDD=INP1
```

The sample job and parameter are customized for a DB2 subsystem name equal to DSNZ and the test environment is DB2 V5 in an OS/390 V8 environment.

$DB2UN00 REXX EXEC

```
/* REXX */
   /*-                  Data Tool for DB managment                    -*/
   /*- Archive DataBase & Migrate Data with Unload External function -*/
arg parmin ; parm = translate(parmin,' ',',')
nparm   = words(parm) ; Subsys  = word(parm,1) ; DBname  = word(parm,2)
Creunl  = word(parm,3); Tsubsys = word(parm,4) ; TDBname = word(parm,5)
Creloa  = word(parm,6); ReusVS  = word(parm,7) ; Ownerb  = word(parm,8)
Full    = word(parm,9); Maxdd   = word(parm,1Ø);KeepTime= word(parm,11)
Autosub = word(parm,12);Jobname = word(parm,13);Catnam  = word(parm,14)
   /*—  Test input parameters               —*/
 if nparm < 14 then do
    say '' ; say '' ; say '>>>>>>>>'
    say '>>>>>>>>  Parameter string is incomplete !!!!'
    say '>>>>>>>>           'parmin
    say '>>>>>>>>' ; say '' ; say '' ; exit ; end
 if Full = '*'  then Full = no
 if Full ¬= no & Full ¬= yes then do
    say '' ; say '' ; say '>>>>>>>>'
    say '>>>>>>>> Wrong FULL 'full' variable !!!!!'
    say '>>>>>>>> Specify : */NO/YES              '
    say '>>>>>>>>' ; say '' ; say '' ; exit ; end
```

```
if ReusVS = '*' then ReusVS = yes
if ReusVS ¬= no & ReusVS ¬= yes then do
   say '' ; say '' ; say '>>>>>>>>'
   say '>>>>>>>> Wrong ReusVS ReusVS' variable  !!!!!'
   say '>>>>>>>> Specify : */NO/YES              '
   say '>>>>>>>>' ; say '' ; say '' ; exit ;end
if Full = yes & Ownerb = '*' then do
   say '' ; say '' ; say '>>>>>>>>'
   say '>>>>>>>> For Archive (Full=yes) increase Ownerb variable.'
   say '>>>>>>>>' ; say '' ; say '' ; exit ; end
if Maxdd =  '*' then Maxdd =  8Ø
if Maxdd > 8Ø then do
   say '' ; say '' ; say '>>>>>>>>'
   say '>>>>>>>> The maximum number of Unload for a job is 8Ø !!!!!'
   say '>>>>>>>>' ; say '' ; say '' ; exit ; end
if Keeptime = '*' then Keeptime = MØ1
if Keeptime ¬= MØ1 & Keeptime ¬= MØ3 & ,
   Keeptime ¬= MØ6 & Keeptime ¬= M12 & ,
   Keeptime ¬= M24 & Keeptime ¬= M36 & ,
   Keeptime ¬= M48 & Keeptime ¬= M6Ø then do
   say '' ; say '' ; say '>>>>>>>>'
   say '>>>>>>>> Wrong KeepTime vaiable !!!!!!  '
   say '>>>>>>>>       specify : */MØ1/MØ3/MØ6/M12/M24/M36/M48/M6Ø'
   say '>>>>>>>>' ; say '' ; say '' ; exit ; end
if (Creunl ¬= '*' & Creloa  = '*') | ,
   (Creunl  = '*' & Creloa ¬= '*') then do
   say '' ; say '' ; say '>>>>>>>>'
   say '>>>>>>>> Unload and Load creator must be both given for the  '
   say '>>>>>>>> update of creator in sysounch datasets              '
   say '>>>>>>>>' ; say '' ; say '' ; exit ; end
if (Creunl ¬= '*' & Creloa ¬= '*') & ,
   (Creunl  = Creloa) then do
   say '' ; say '' ; say '>>>>>>>>'
   say '>>>>>>>> Creunl and Creloa must be different.'
   say '>>>>>>>>' ; say '' ; say '' ; exit ; end
if Autosub = '*' then Autosub = no
if Autosub = yes & Full = no then do
   say '' ; say '' ; say '>>>>>>>>'
   say '>>>>>>>> Full and Autosub parameter are incompatible !!!!'
   say '>>>>>>>>' ; say '' ; say '' ; exit ; end
if Autosub ¬= no & Autosub ¬= yes then do
   say '' ; say '' ; say '>>>>>>>>'
   say '>>>>>>>> Wrong Autosub variable !!!!!'
   say '>>>>>>>> Specify : */YES/NO             '
   say '>>>>>>>>' ; say '' ; say '' ; exit ; end
if TDBname = '*' then TDBname = DBname
if Jobname  = '*' then do
   Jobname  = substr(DBname,1,2)||substr(DBname,4,4)
   Jobname  = translate(Jobname,'$',' ')
   TJobname = substr(TDBname,1,2)||substr(TDBname,4,4)
   TJobname = translate(TJobname,'$',' ') ;  end
else do
```

```
    lJobname = length(Jobname)
    if lJobname > 6 then do
     say '' ; say '' ; say '>>>>>>>>'
     say '>>>>>>>> the Jobname variable may be max six caracters long!!'
     say '>>>>>>>>' ; say '' ; say '' ; exit ; end ; end
 if catnam = '*' then catnam = SYSIBM
 if Tsubsys = '*' then Tsubsys = Subsys
   /*-  Parameters assignment                -*/
 call @db2par0 Subsys ; if word(result,1) = 99 then exit
 $lpar =word(result,1) ; $accn =word(result,2) ; $class =word(result,3)
 $msgcla = word(result,4);$region  = word(result,5) ;$msglvl =
word(result,6)
 $notif  = word(result,7);$user    = word(result,8) ;$unitda =
word(result,9)
 $unitta = word(result,10);$esunit = word(result,11);$prt    =
word(result,12)
 $hiwork = word(result,13);$db2ver = word(result,14);$ctsubs =
word(result,15)
 $librexx= word(result,16);$parmlib= word(result,17);$proclib=
word(result,18)
 $jcllib = word(result,19);$report = word(result,20);$libexec=
word(result,21)
 $isptenu= word(result,22);$isppenu= word(result,23);$ispmenu=
word(result,24)
 $ispslib= word(result,25);$plilink= word(result,26);$sibmlnk=
word(result,27)
 $sortlib= word(result,28);$hilvlDB= word(result,29);$runlib =
word(result,30)
 $dsnload= word(result,31);$tep2pgm= word(result,32);$tep2pln=
word(result,33)
 $unlopgm= word(result,34);$unlopln= word(result,35);$dunlopg=
word(result,36)
 $dunlopl= word(result,37);$dsnproc= word(result,38)
   /*-   Work areas initialization          -*/
 blk    =    ;okunlo = off ;stp    =  0 ;fullstp = 0 ;njob    =  1
 njob1  = 00 ;njobRO = 00  ;njobRW = 00 ;ctst0   = 0 ;testata = yes
 ttracks=  0 ;ctrsts =  0  ;tcyls  =  0 ;#c    = 0 ;#x      = 0 ;#y =  0
 #p1    =  7 ; jna = '&SYSUID‘ ;$notif = '&SYSUID‘
 iebn = IE||substr(DBname,1,2)||substr(DBname,4,2)
 data = substr(date(s),7,2)‘/‘substr(date(s),5,2)‘/‘substr(date(s),1,4)
 ora  = time()
   /*-  SYSTSIN file allocation             -*/
  say '' ; outdstsin= $hiwork'.'subsys'.'DBname'.@DB2UN00.SYSTSIN'
  prmalloc = subsys' 'outdstsin' 0 5,1 f,b 80 27920 systsinp yes'
  call @db2all0 prmalloc ; if word(result,1) = 99 then exit
sk.1='DSN SYSTEM('subsys')                                          '
sk.2='RUN PROGRAM('$dunlopg') PLAN('$dunlopl') LIB('''$runlib''') -   '
sk.3='LIB('''$runlib''') PARM(''SQL'')                              '
sk.4='END                                                           '
 sk.0=4 ; jobw = systsinp ; call WriteRec
   /*-  SYSPRINT file allocation            -*/
```

```
 outdsprt= $hiwork'.'subsys'.'DBname'.@DB2UN00.SYSPRINT'
 prmalloc = subsys' 'outdsprt' 0 1,15 f,b,a 121 1210 sysprint yes'
 call @db2all0 prmalloc ; if word(result,1) = 99 then exit
   /*-  SYSREC00 file allocation            -*/
 outdsrec= $hiwork'.'subsys'.'DBname'.@DB2UN00.SYSREC00'
 prmalloc = subsys' 'outdsrec' 0 15,45 f,b 0 0 sysrec00 yes'
 call @db2all0 prmalloc ; if word(result,1) = 99 then exit
   /*-  SYSPUNCH dummy file allocation      -*/
 "alloc dummy f(syspunch)"
   /*-  SYSIN file allocation               -*/
 outdsin= $hiwork'.'subsys'.'DBname'.@DB2UN00.SYSIN'
 prmalloc = subsys' 'outdsin' 0 5,1 f,b 80 2720 sysin yes'
 call @db2all0 prmalloc ; if word(result,1) = 99 then exit
sk.1=' SELECT CREATOR,NAME,CARD,                                        '
sk.2='        SUM(CARD*(6+RECLENGTH)),                                  '
sk.3='        (CURRENT TIMESTAMP - STATSTIME),                          '
sk.4='        NTABLES,TSNAME,VCATNAME,PARTITIONS,ENCODING_SCHEME,       '
sk.5='        DBID,PSID,OBID,STATSTIME,CREATEDTS,ALTEREDTS              '
sk.6='        FROM V'DBname'_UNLOAD                                     '
sk.7=' GROUP BY CREATOR,NAME,NTABLES,TSNAME,VCATNAME,STATSTIME,         '
sk.8='          CARD,PARTITIONS,ENCODING_SCHEME,DBID,PSID,OBID,         '
sk.9='          CREATEDTS,ALTEREDTS                                     '
sk.10=' ORDER BY NTABLES DESC,TSNAME,NAME  ;                            '
 sk.0=10; jobw = sysin ; call WriteRec
   /*-  Call DB2                           -*/
 xx=outtrap(trpdb20.) ; address tso "ex '"outdstsin"'" ; xx=outtrap(off)
   /*-  Print DB2 output                    -*/
 xx=outtrap(trpdummy.) ; address tso "printoff ('"outdsprt"')  class(R)"
 xx=outtrap(off)
 if rc > 0 then do
    analisi = substr(trpdb20.1,1,8)
    select
   /*-  DB2 subsystem not active           -*/
       when analisi = 'DSNE100I' then do
          subs = center(subsys,6)
         say '>>>>>>>>'  ; say '>>>>>>>>'
         say '>>>>>>>>   The DB2 subsystem ->'subs'<- is not active !!!'
         say '>>>>>>>>' ; say '>>>>>>>>'
         say '' ; say '' ; call Free0 ; exit ; end
   /*-  DB2 subsystem not defined on LPAR  -*/
       when analisi = 'DSNE110E' then do
          subs = center(subsys,6) ; say '>>>>>>>>' ; say '>>>>>>>>'
  say '>>>>>>>>  The DB2 subsystem -> 'subs' <- is not defined on' $lpar
          say '>>>>>>>>' ; say '>>>>>>>>'
          say '' ; say ''  ; call Free0 ; exit ; end
       otherwise
          say '>>>>>>>>' ; say '>>>>>>>>'
          say '>>>>>>>>  Unpredictable error !!!!!                '
          say '>>>>>>>>            Contact your Support staff      '
          say '>>>>>>>>' ; say '>>>>>>>>'
          DO #a = 1 to trpdb20.0 ; say trpdb20.#a ; end
```

```
         exit ; end ; end
 else do
   /*-  Start process                          -*/
    xx=outtrap(trpread00.)
       "execio * diskr sysprint (stem sysprint. finis"
    xx=outtrap(off)
   /*-  Verify Select                          -*/
    do #b = 1 to sysprint.0
       okunlo = word(sysprint.#b,2)
       if okunlo = 'DSNT495I' then do
          okunlo = yes ; okrec = word(sysprint.#b,5) ; end ; end
   /*-  DB2 access OK Start process           -*/
    if okunlo = yes then do
       if okrec > 0 then do ; call Alloc00 ; call Elabo00 ; end
       else do
          say '' ; say '>>>>>>>>' ; say '>>>>>>>>'
          say '>>>>>>>>    0 Record found. ||||||  '
          say '>>>>>>>>' ; say '>>>>>>>>'
          say '' ; call Free0 ; exit ; end ; end
    else do
       say '' ; say '' ; say '>>>>>>>>' ; say '>>>>>>>>'
       say ' DB2 access KO. Verify the sysout '$user' for analyse the
failure reason !!'
       say '>>>>>>>>' ; say '>>>>>>>>' ; say ''
       do #b = 1 to sysprint.0 ; say sysprint.#b ; end
       call Free0 ; end ; end ; call Free
 exit
 Elabo00 :
   /*-  Read extracted Records              -*/
    xx=outtrap(trpread01.)
       "execio * diskr sysrec00 (stem sysrec00. finis"
    xx=outtrap(off)
    if rc > 0 then do
       do #a = 1 to trpread01.0 ; say trpread01.#a ; end
       say ''  ; say '' ; say '>>>>>>>>'
       say '>>>>>>>>   Error reading file "'outdsrec'"    '
       say '>>>>>>>>   RC='rc'. Verify.                   '
       say '>>>>>>>>' ; say '' ; say '' ; exit ; end
    say ''
    say '  ********************************************************  '
    say '*                                          'time()'     *'
    say '*        building DB2 procedure                          *'
    say '*                                                        *'
    say '*                        wait please ......              *'
    say '*                                                        *'
    say '  ********************************************************  '
    say ''
    welab25 = (sysrec00.0 * 25)%100 ; welab50 = (sysrec00.0 * 50)%100
    welab75 = (sysrec00.0 * 75)%100 ; welab00 = sysrec00.0
    DO #d = 1 to sysrec00.0
       trkp = 0 ; trks = 0 ; notrat = sysrec00.0 ; stp = stp + 1
       fullstp= fullstp + 1
```

```
       $ntable = x2d(d2x(c2d(substr(sysrec00.#d,49,2))))
       $tsname = strip(substr(sysrec00.#d,51,8))
   /*-  Build dsname routine                  -*/
       nome = blk ; call @db2tbna $tsname
       if word(result,1) = 99 then exit
       nome = word(result,1)
   /*--  write header job                     -*/
       if testata = yes then do ; ctst0 = ctst0 + 1
sk.1='//'Jobname||njob1' JOB ('$accn'),''Unload Dati'',CLASS='$class','
sk.2='//      MSGCLASS='$msgcla',USER='$user',REGION='$region',       '
sk.3='//      MSGLEVEL=('$msglvl'),NOTIFY='$notif
sk.4='/*JOBPARM BYTES=999999,LINES=9999                               '
sk.5='//* ------------------------------------------ *               '
sk.6='//* ---------   U n l o a d     D a t a   ------- *             '
sk.7='//* ------------------------------------------ *               '
sk.8='//JOBLIB    DD  DISP=SHR,DSN='$dsnload
sk.9='//DB2PROC   JCLLIB ORDER=('$proclib')                           '
          sk.0=9
          if Full = yes then do
sk.10='//* ------------------------------------------ *              '
sk.11='//STRRO'njob1'  EXEC PGM=IKJEFT01,DYNAMNBR=20                  '
sk.12='//SYSTSPRT  DD  SYSOUT=*                                       '
sk.13='//SYSTSIN   DD  DISP=SHR,                                      '
sk.14='//          DSN='outdsrep'(STRRO'njob1')                       '
             sk.0=14 ; end
          jobw = FIUNL ; "alloc da('"outdsunl"') f("jobw") mod reuse"
          Call WriteRec
sk.1='//'TJobname||njob1' JOB ('$accn'),''Load Dati'',CLASS='$class', '
sk.2='//      MSGCLASS='$msgcla',USER='$user',REGION='$region',       '
sk.3='//      MSGLEVEL=('$msglvl'),NOTIFY='$notif
sk.4='/*JOBPARM BYTES=999999,LINES=9999                               '
sk.5='//* ------------------------------------------ *               '
sk.6='//* -------      L o a d    D a t a     --------- *             '
sk.7='/* ------------------------------------------ *                '
sk.8='//JOBLIB    DD  DISP=SHR,DSN='$dsnload
sk.9='//DB2PROC   JCLLIB ORDER=('$proclib')                           '
          sk.0=9
          jobw = FILOA ; "alloc da('"outdsloa"') f("jobw") mod reuse"
          Call WriteRec
sk.1='//'$user||right(njob1,1)' JOB ('$accn'),''Del/Wrk/
Areas'',CLASS='$class','
sk.2='//      MSGCLASS='$msgcla',USER='$user',REGION='$region',       '
sk.3='//      MSGLEVEL=('$msglvl'),NOTIFY='$notif
sk.4='/*JOBPARM BYTES=999999,LINES=9999                               '
sk.5='//* ------------------------------------------ *               '
sk.6='//* -------   Delete  data-set  di  W o r k   --- *             '
sk.7='//* ------------------------------------------ *               '
sk.8='//DELWORK  EXEC PGM=IDCAMS                                      '
sk.9='//SYSPRINT  DD SYSOUT=X                                         '
sk.10='//SYSIN     DD  *                                              '
          sk.0=10
          jobw = FIDEL ; "alloc da('"outdsdel"') f("jobw") mod reuse"
```

```
          Call WriteRec ; jnamod = iebn || njob1
sk.1='//'jnamod' JOB ('$accn'),''Copy Syspunch'',CLASS='$class',      '
sk.2='//      MSGCLASS='$msgcla',USER='$user',REGION='$region',       '
          if ctstØ = 1  then
sk.3='//      MSGLEVEL=('$msglvl'),NOTIFY='$notif
          else
sk.3='//      MSGLEVEL=('$msglvl'),NOTIFY='$notif',TYPRUN=HOLD        '
sk.4='/*JOBPARM BYTES=999999,LINES=9999                               '
sk.5='//* ------------------------------------------- *               '
sk.6='//* ---  Iebgener disp=mod for syspunch copy  --- *               '
sk.7='//* ------------------------------------------- *               '
sk.8='//DB2PROC   JCLLIB ORDER=('$proclib')                           '
sk.9='//* ------------------------------------------- *               '
          if ctstØ = 1 & Full = yes then do
sk.1Ø='//STEPØØ   EXEC   DB2REXX1                                     '
sk.11='//REXXØØ.SYSTSIN  DD  *                                        '
sk.12='  ISPSTART CMD(@DB2RCØØ 'subsys','DBname','xxxxxx',BLK,@DB2UNØØ)'
sk.13='//* ------------------------------------------- *               '
             #y = 13 ; end
          else #y = 9
          #y = #y + 1
sk.#y='//COPY    PROC LIB=,DISP=MOD                                   '
          #y = #y + 1
sk.#y='//COPY1   EXEC PGM=IEBGENER                                    '
          #y = #y + 1
sk.#y='//SYSPRINT  DD  SYSOUT=*                                       '
          #y = #y + 1
sk.#y='//SYSUT1    DD  DSN=&LIB,DISP=SHR                              '
          #y = #y + 1
sk.#y='//SYSUT2    DD  DSN='outdsmod',                                '
          #y = #y + 1
sk.#y='//          DISP=&DISP                                         '
          #y = #y + 1
sk.#y='//SYSIN     DD  DUMMY                                          '
          #y = #y + 1
sk.#y='//    PEND                                                     '
          sk.Ø=#y
          jobw = FIIEB ; "alloc da('"outdsieb"') f("jobw") mod reuse"
          Call WriteRec ; testata = 'NO'
          if Full = yes then do
   /*—  input archive output Unload         —*/
sk.1='  ISPSTART CMD+                                                 '
sk.2='(@DB2UNØ3 'outdsrep',UNLO'right(njob,4,'Ø')','Jobname||njob1')  '
             sk.Ø=2
             jobw = FISOU ; "alloc da('"outdssou"') f("jobw") mod reuse"
             Call WriteRec ; end ; end
   /*—  Step  U N L O A D                   —*/
       utilid = U||DBname||substr(time('L'),1Ø,6)
       fullstp = right(fullstp,5,'Ø')
sk.1='//* --------------- 'fullstp' ---------------------- *         '
       if Full = yes & stp = 1 then do
sk.2='//LABØ     IF (STRRO'njob1'.RC EQ Ø) THEN                       '
```

```
          #y = 2
          end
       else
          #y = 1 ; #y = #y + 1
sk.#y='//UNLO'stp'    EXEC '$dsnproc',PARM='''subsys','utilid''''
          #y = #y + 1
sk.#y='//SYSPUNCH  DD  DSN='$hiwork'.'DBname'.'nome'.PUNCH,           '
          #y = #y + 1
sk.#y='//          UNIT='$esunit',DISP=(,CATLG),SPACE=(TRK,(5,1),RLSE)'
          #y = #y + 1
sk.#y='//SYSREC00  DD  DSN='$hiwork'.'DBname'.'nome',                 '
   /*-  Fill-up Tablespace information       -*/
       do #e = 1 to $ntable
          $creator   = strip(substr(sysrec00.#d,1,8))
          $tblname   = strip(substr(sysrec00.#d,11,18))
          $tsname    = strip(substr(sysrec00.#d,51,8))
          $vcatname  = strip(substr(sysrec00.#d,59,8))
          $partition = x2d(d2x(c2d(substr(sysrec00.#d,67,2))))
          $encoding  = strip(substr(sysrec00.#d,69,1))
          $dbid      = x2d(d2x(c2d(substr(sysrec00.#d,70,2))))
          $psid      = x2d(d2x(c2d(substr(sysrec00.#d,72,2))))
          $obid      = x2d(d2x(c2d(substr(sysrec00.#d,74,2))))
          $statstime = strip(substr(sysrec00.#d,76,26))
          $createdts = strip(substr(sysrec00.#d,102,26))
          $alteredts = strip(substr(sysrec00.#d,128,26))
          $card      = c2x(substr(sysrec00.#d,29,4))
          if $card  ¬= 'FFFFFFFF' then do
             $card     = x2d(d2x(c2d(substr(sysrec00.#d,29,4))))
             $cardrecl = x2d(d2x(c2d(substr(sysrec00.#d,33,4))))
             $diffstat = d2x(c2d(substr(sysrec00.#d,37,6)))
             lstats    = length($diffstat)
             end
          else do
             $card = -1 ; $cardrecl = 0 ; $diffstat = 10
             lstats = length($diffstat) ; end
          if $card = -1 then do
             #c = #c + 1 ; wr2 = left($creator'.'$tblname,27,' ')
             jk.#c='>>>>>>>> 'wr2' has no CARD value ||||| ' ;  end
/*impl00*/
   /*-  Compute SYSREC00 space               -*/
          wtrk = $cardrecl % 51000
          if wtrk < 15 then wtrk = 15
          trkp = trkp + wtrk ; #d = #d + 1 ; call VerElab
   /*-  Runstats older than one day          -*/
          if $diffstat > 9 then ctrsts = ctrsts + 1
          end  /* DO #e End */
       #d = #d - 1  ; trks = trkp % 10
       ttracks = ttracks + trkp + 5 ; #y = #y + 1
sk.#y='//
UNIT='$esunit',DISP=(,CATLG),SPACE=(TRK,('trkp','trks'),RLSE)
       #y = #y + 1
sk.#y='//SYSIN     DD  *                                              '
```

```
       #y = #y + 1
sk.#y='  REORG TABLESPACE 'DBname'.'$tsname' LOG NO UNLDDN SYSREC00   '
       #y = #y + 1
sk.#y='      UNLOAD EXTERNAL                                          '
       #y = #y + 1
sk.#y='//* ------------------------------------------- *             '
       #y = #y + 1
sk.#y='//UPDATE'stp' EXEC   DB2REXX1                                  '
       #y = #y + 1
sk.#y='//REXX00.SYSTSIN  DD  *                                        '
       #y = #y + 1
sk.#y=' ISPSTART CMD(@DB2UN01 +                                       '
       #y = #y + 1
sk.#y=' 'subsys','DBname'.'nome','Creunl','Creloa','ReusVS')          '
       if stp = maxdd | Fullstp = sysrec00.0 then do
          #y = #y + 1
sk.#y='//* ------------------------------------------- *           '
          end ; sk.0=#y
       jobw = FIUNL ; "alloc da('"outdsunl"') f("jobw") mod reuse"
       Call WriteRec
   /*-  Step  L O A D                          -*/
       utilid = L||DBname||substr(time('L'),10,6)
sk.1='//* --------------- 'fullstp' ----------------------- *         '
sk.2='//LOAD'stp'    EXEC '$dsnproc',PARM='''Tsubsys','utilid''''
sk.3='//SYSREC00  DD  DISP=SHR,                                       '
sk.4='//          DSN='$hiwork'.'DBname'.'nome
sk.5='//SYSIN     DD  DISP=SHR,                                       '
sk.6='//          DSN='$hiwork'.'DBname'.'nome'.PUNCH                 '
       sk.0=6
       if stp = maxdd then do
sk.7='//* ------------------------------------------- *             '
          sk.0=7 ; end
       jobw = FILOA ; "alloc da('"outdsloa"') f("jobw") mod reuse"
       Call WriteRec
   /*-------  Step  iebgener disp=mod            -------*/
       select
          when fullstp = 1 then do
sk.1='//COPY'stp' EXEC COPY,LIB='$hiwork'.'DBname'.'nome'.PUNCH,      '
sk.2='//           DISP='SHR'                                         '
             end
          when #d = sysrec00.0 then do
sk.1='//COPY'stp' EXEC COPY,LIB='$hiwork'.'DBname'.'nome'.PUNCH,      '
sk.2='//           DISP=''MOD,SPACE=(CYL,(0,0),RLSE)''                '
             end
          otherwise
```

Editor's note: the code for this article will be concluded in the next issue.

Giuseppe Rendano
DB2 Systems Programmer (Italy)

Explaining the UDB DB2 Governor

This article discusses running the UDB DB2 Governor (V7.2) on mid-range systems (NT and Unix). It covers mainly Enterprise Edition (EE) systems, but discusses what you need to do to run the Governor on an Extended Enterprise Edition (EEE) system.

WHY USE A GOVERNOR?

The main reason for using a Governor is to stop runaway processes/ extremely long-running queries from grinding your system to a halt. The DB2 Governor is the ideal tool for this. One thing the DB2 Governor cannot do is educate end users! It should only be implemented in association with end user education, otherwise you will hit the common problem of users submitting the same query two, three, or even four times when their original request is 'governed', making any performance problems even worse!

HOW DOES THE DB2 GOVERNOR WORK?

The DB2 Governor comes bundled together with UDB DB2. It is a reactive tool not a pro-active one. What I mean by this is that it does not look at a piece of SQL and determine how long it will take to run, how many rows it will read, etc, and then decide whether the SQL exceeds certain limits, but, rather, it monitors executing SQL to see whether it exceeds any boundaries that you have specified. These boundaries can be based on CPU time used (Unix systems only), rows read, rows selected, number of locks taken, idle time, and UOW time. They can be made to apply to application names and/or userids (not groups), and you also define what action is to take place if any of these boundaries are crossed.

ARE THERE ANY PRE-REQUISITES?

On NT, before you can start the DB2 Governor task, you have to make sure that the DB2 Governor service is installed. You can tell if it is already installed by going to **Start/Settings/Control Panel/Services**.

If you can see a DB2 Governor process, then it is installed. Otherwise, you need to install the DB2 Governor service as follows:

```
<x>:\>db2reggv i
ok
```

where <x> is the drive you installed DB2 on.

(>db2reggv u will un-register the DB2 Governor service.)

The db2reggv program is in the *<x>/Program Files/sqllib/bin* directory.

On Unix, there isn't a DB2 Governor service that you have to install before using the DB2 Governor task.

HOW TO START THE DB2 GOVERNOR TASK

The DB2 Governor task needs a configuration file to start from, and needs to write information to a log file. Both of these files have to be specified at DB2 Governor start-up time. Assume that a configuration file (samp01.txt) exists in the <x>:\db2gov directory on an NT system, and you want to start the DB2 Governor for the sample database, then the command to start the DB2 Governor is:

```
>cd <x>:\db2gov
><x>:\db2gov>db2gov start sample samp01.txt ..\..\..\..\db2gov\log.txt
```

Or, if you are not in the *<x>:\db2gov* directory, you can issue:

```
><x>:>db2gov start sample <x>:\db2gov\samp01.txt
..\..\..\..\db2gov\log.txt
```

On NT, the default file location for the logs is *x:\Program Files\sqllib\log*.

On Unix systems the command to start the DB2 Governor would be (assuming you have created the config01 file in the db2gov directory, which you also need to create):

```
$cd /home/db2inst1/db2gov
$ db2gov start sample config01    /../../db2gov/log3
db2gov: Starting db2govd for database sample
```

(Because the default path for the log is */home/<instance-name>/sqllib/log*.)

On a Unix EEE system, the command to start the DB2 Governor would be:

```
/home/db2inst1/db2gov>db2gov start <database> nodenum <num> <config> <log>
```

where:

- <*database*> is the database name.
- <*num*> is the node number on which to start the DB2 Governor. This is the node number in the home/<*instance-name*>/sqllib/db2nodes.cfg file.
- <*config*> is the configuration file.
- <*log*> is the log file. The number of the database partition is appended to the log file name (see example below).

For example, suppose we have a database sample, spread over three nodes (0 to 2), in the db2inst1 instance, then, to start the DB2 Governor, issue:

```
/home/db2inst1/db2gov$db2gov start sample config01 /../../db2gov/log4

sysprod: db2gov: Starting db2govd for database sample on node 0
sysprod: db2gov: Validating configuration file
sysprod: db2gov start sample ... completed ok

sysprod: db2gov: Starting db2govd for database sample on node 1
sysprod: db2gov: Validating configuration file
sysprod: db2gov start sample ... completed ok

sysprod: db2gov: Starting db2govd for database sample on node 2
sysprod: db2gov: Validating configuration file
sysprod: db2gov start sample ... completed ok
```

The above command will write the log files to the db2gov directory, which can be seen if you do an **ls** command:

```
/home/db2inst1/db2gov$ls
config01  log4.0    log4.1    log4.2
```

(Note the partition suffix on each of the log files.)

To stop the DB2 Governor on an EEE system, use the usual command:

```
/home/db2inst1/db2gov$db2gov stop sample
sysprod: db2gov: Stopping db2govd for database sample on node 0
sysprod: db2gov stop sample nodenum ... completed ok
```

```
sysprod: db2gov: Stopping db2govd for database sample on node 1
sysprod: db2gov stop sample nodenum ... completed ok

sysprod: db2gov: Stopping db2govd for database sample on node 2
sysprod: db2gov stop sample nodenum ... completed ok
```

To just start the DB2 Governor on one node:

```
/home/db2inst1/db2gov$db2gov start sample nodenum 2 config01 /../../
db2gov/log4
db2gov: Starting db2govd for database sample on node 2
db2gov: Validating configuration file
```

If you start the DB2 Governor on only one node, then it operates on only that one node, ie only SQL run from that node will be governed. To ensure that you were running on that node, you would have to issue an EXPORT DB2NODE command, as shown below:

```
$export DB2NODE=2
$db2 terminate
$db2 <command>
```

You stop it in the usual way:

```
/home/db2inst1/db2gov$db2gov stop sample
```

HOW CAN YOU TELL WHETHER THE DB2 GOVERNOR IS ACTIVE

The only way I have found to tell whether the DB2 Governor is active is to try to start it again. If it is already running you will get the following message:

```
db2govd: GOV1007N  Governor already flagged as running.  If it is not
running, use  'db2gov stop' to clean up.
```

HOW TO STOP THE DB2 GOVERNOR

To stop the DB2 Governor:

```
$db2gov stop <database-alias>
```

This applies to both NT and Unix EE and EEE systems.

You don’t need to be in the directory containing the configuration file to stop the DB2 Governor.

DB2 GOVERNOR PROCESSING

When the DB2 Governor task is invoked, the first thing it does is to see whether the configuration has changed since the last time it was read. In this way you can make changes to the configuration file without having to stop/start the DB2 Governor.

The DB2 Governor uses a snapshot to get the information about all running applications. Once the DB2 Governor has this snapshot information, it compares it with the values in the configuration file and, if a condition is met, takes whatever action is necessary. Once it has finished processing, it sleeps for the time indicated in the configuration file (interval), wakes up, and starts the process all over again.

DB2 GOVERNOR LOGGING

The DB2 Governor logs every action it takes to the log file specified at start-up time. If there was an error in the configuration file, this is written back to the screen from where you issued the start command.

WHAT DOES THE DB2 GOVERNOR LOG CONTAIN?

On start-up you will see the following entries in the DB2 Governor log:

```
$ cat log3
2001-07-15-21.33.14    0 START      Database = SAMPLE
2001-07-15-21.33.14    0 READCFG    Config = /home/db2inst1/db2gov/
config01
```

Every time a condition is tripped, an entry is written to the log:

```
2001-07-15-21.37.22    0 FORCE      applname 'db2bp' authid 'HARRY'
applid '*LOCAL.db2inst1.011015133720' coord 0 (line 5)  Rowsread 8
2001-07-15-21.37.23    0 ACCOUNT    PAVKC
*LOCAL.db2inst1.011015
33720     db2bp                2001-07-15-21.37.20          0          0
```

WHAT CONFIGURATION FILE ARE YOU RUNNING FROM?

To find out what configuration file you are running from, look in the DB2 Governor log file and search for READCFG. You will see a line like:

```
yyyy-mm-dd-hh.mm.ss    0 READCFG    Config = C:\db2gov\samp01.txt
```

The admin guide gives a full description of what the configuration file should contain. The important thing to remember is that the applname is case sensitive (everything else isn't). An example of a configuration file is:

```
interval 1;
dbname SAMPLE;
account 10;

desc"This is rule1"
time 00:00 23:59
authid user01
setlimit
cpu -1
rowsread 3
rowssel -1
locks -1
idle -1
uowtime -1
action force ;

desc"This is rule2"
. . . .
. . . .
```

The first three lines of the configuration file define which database the sections apply to, how often the DB2 Governor task should wake up and process the configuration file (interval – default 120 seconds), and when accounting records are written (account – no default). The interval value is specified in seconds, and the account value is specified in minutes. The account value cannot be less than the interval value. A snapshot is taken at every interval.

Resource limits can be set to apply to:

- time (certain time periods,00:00 to 23:59) – if you do not specify anything, then the rule is valid for 24 hours.
- authid (certain users) – separate multiple authids with a comma. If authid does not appear, then the rule applies to all authids. You cannot specify groups.
- applname (certain applications) – specifies the name of the .exe file that makes the connection to the database. Separate multiple

names with a comma. If the clause does not appear, then the rule applies to all applications. This is case-sensitive.

Resource limits may be set on:

- cpu (CPU time) – total number of CPU seconds that the application can use (not available on Windows).
- rowsread (rows read) – total number of rows read to satisfy the query (including catalog reads etc).
- rowssel (rows selected) – total number of rows returned to the application.
- locks (number of locks held) – total number of locks the application can hold.
- idle (connect time) – the number of idle seconds allowed for a connection before an action is taken.
- uowtime (elapsed time) – total number of seconds that the unit of work can run for.

For each of the above, if you put '-1' (without the quotes), then there is no limit on that particular parameter.

Action can be:

- Force – to force the agent that is serving the application.
- Priority – can be –20 to +20 (but not +0 – if you want zero, then put just 0).
- Schedule – to change the scheduling priority of the user.

If you put the action as priority 0 (action priority 0;) then nothing happens to any offending query, but the fact that the query exceeded a limit in the configuration file is recorded in the logs. This allows you to build up a historical picture, about which queries are selecting the most rows perhaps, or are taking excessive amounts of locks.

Comments should be enclosed within curly brackets, {}.

In an EEE environment, the configuration file must be able to be seen from each machine.

What happens if more than one rule applies to an application?

You can have as many rules as you like in the configuration file, and several can monitor the same application. This situation can get very messy. Basically, the rule furthest down the file is the rule that will be used (providing the limit is not set to –1). If it is, then the values in previous rules are used in combination with the last rule. This is all explained in the admin guide, but I would recommend that, where possible, you have only one rule per application.

THE DB2 GOVERNOR LOG ANALYZER

To look at what the DB2 Governor has been doing, you can either look at the log file directly, or you can use the db2govlg utility. The command is as follows:

```
$db2govlg <log-file-name> rectype <record-type>
```

For example:

```
c:\user>db2govlg ..\..\..\..\db2gov\log8.txt
```

(The above assumes that the log is in the C:\db2gov directory. Don't forget that the default log directory on NT is *C:\PROGRAM FILES\SQLLIB\DB2\log*, so you have to point it to the *c:\db2gov* directory.)

The different <*record type*> values are:

- STARTe – the DB2 Governor was started.
- READCFGe – the configuration file was read.
- STOPe – the DB2 Governor was stopped.
- FORCEe – an application was forced.
- NICEe – the priority of an application was changed.
- ERROREe – basically means the DB2 Governor has stopped.
- WARNING
- ACCOUNTe – indicates what area broke a CPU usage rule.

To just see when the configuration file was read, you would issue (assuming that the log file was called log8 in the *c:\db2gov* directory):

```
C:\db2gov>db2govlg ..\..\..\..\db2gov\log8.txt rectype readcfg
2001-07-01-19.43.02    0 READCFG    Config = C:\db2gov\samp01.txt
2001-07-01-19.44.50    0 READCFG    Config = C:\db2gov\samp01.txt
2001-07-01-19.45.37    0 READCFG    Config = C:\db2gov\samp01.txt
2001-07-01-19.50.36    0 READCFG    Config = C:\db2gov\samp01.txt
2001-07-01-19.58.29    0 READCFG    Config = C:\db2gov\samp01.txt
2001-07-01-20.01.34    0 READCFG    Config = C:\db2gov\samp01.txt
2001-07-01-20.10.37    0 READCFG    Config = C:\db2gov\samp01.txt
```

If you want to see when/why a user was forced off, then issue:

```
C:\db2gov>db2govlg ..\..\..\..\db2gov\log8.txt rectype force
2001-07-01-19.49.06    0 FORCE      applname 'db2bp.exe' authid 'USER01'
applid '*LOCAL.DB2.011001184903' coord 0 (line 4)  Rowsread 8
2001-07-01-20.01.34    0 FORCE      applname 'db2bp.exe' authid 'USER01'
applid '*LOCAL.DB2.011001185042' coord 0 (line 4)  Rowsread 883
```

You can store the information from the db2govlg utility in DB2 tables. On Unix systems you can pipe the output from the utility to DB2 tables. On NT you can write a script to read the output and insert the rows into the DB2 tables. This to me seems the only reason to use the log analyzer rather than just browse the log file!

DEPLOYMENT STRATEGY

If you want to capture historical data, then the DB2 Governor should be stopped/started daily at midnight, and the log file renamed to contain the date stamp. If you put the action as ‘priority 0’, then you can see what violations occurred without actually impacting any users.

POSSIBLE ERROR MESSAGES

On NT you may see the error message: GOV1062N. It could mean that the Governor daemon is not started!

```
C:\db2gov>db2gov start sample sam01.cfg samlog.txt
OpenEvent1:The system cannot find the file specified.

db2gov: GOV1062N  Unable to start governor at node "0". rc = "2".
```

If you get this message (and the config file is there!), make sure that the DB2 Governor daemon is in service. If it isn't, you have to register the DB2 Governor using the db2reggv command.

CONCLUSION

The DB2 Governor is an excellent tool for controlling resource usage on a system. It should only be introduced in conjunction with user education into the impact on system resources of badly written/long running SQL. Only then will everyone feel that there is a win/win outcome.

FURTHER READING

For further reading see: SC09-2840-00, *Administration Guide – Performance* – Chapter 9.

C Leonard
Freelance Consultant (UK)

Leveraging mainframe data using a data warehouse

The author has written seven articles for *DB2 Update* on using DB2 for data warehousing and believes it is the only practical technology for leveraging mainframe data, particularly if some of that data is in VSAM, IMS, Informix, Oracle, SQL Server, or Sybase files.

Data warehouse is a concept not a product. It is the compiling, assembling, and consolidating of application data common to user communities at a single logical point. Typical uses include *ad hoc* and 'what if' queries, data matching, trend analysis, and other information functions. Warehouse data is generally extracted from OLTP databases that are optimized for transaction processing. Data warehouses are optimized for information processing (see *DB2 Update*, Issue 83, September 1999).

DB2 is a relational database based on the pioneering work of Dr Ted Codd. The key to a successful relational data warehouse is to abandon normalization and use a star schema that minimizes the quantity of joins required to answer queries. Star schemas have two table types of fact and dimension. Fact is at the star centre containing columns to be measured, like sales and units. Dimension is at the star points containing measuring columns, such as date and location.

Star schemas provide better performance, as does using little known features like simple tablespace and setting PCTFREE and FREEPAGE to 0. Simple tablespace allows DBAs to mix rows of multiple tables such as FACT and DIM in a single table thereby avoiding a physical I/O, when doing joins. A PCTFREE and FREEPAGE of 0 reduces table maintenance overhead. This is possible because a data warehouse is read-only (see *DB2 Update*, Issue 86, December 1999).

IBM is a pioneer in data warehousing. It began making extensive improvements when it included OLAP (On Line Analytical Processing) in DB2 UDB V6 with major updates in V7. OLAP provides ranking, row numbering, and existing column function data, as a scalar value in a query result table. OLAP can satisfy a high percentage of user analytical requests without resorting to external software or hardware (see *DB2 Update*, Issue 102, April 2001). V7.1 lets users model their business as a multidimensional cube whose typical dimensions are geography, product, and time. DB2 OLAP Server uses Essbase (Hyperion Solutions Corporation) as its multidimensional engine, but can store data in DB2, allowing it to perform relational management functions such as back-up and recovery. Any OLAP application can be assigned to DB2 using a star schema.

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 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 (see *DB2 Update*, Issue 103, May 2001). DB2's enhanced SQL with new join and GROUP BY features can answer many cube queries without needing multidimensional software (see

DB2 Update: Issue 97, November 2000; Issue 98, December 2000; and Issue 100, February 2001).

IBM added DB2 Warehouse Manager (DB2WM) to DB2 OLAP Server for OS/390 V7.1 (see *DB2 Update,* Issue 105, July 2001). IBM makes DBA tasks easier by:

1 Simplifying prototyping, development, and deployment of a data warehouse.

2 Allowing a data centre to govern queries, analyse costs, manage resources, and track usage.

3 Helping users find, understand, and access their data.

4 Providing more flexible tools and techniques to build, manage, and access the data warehouse.

5 Allowing IBM's expanded SQL to answer common user queries.

DB2WM can get data from lots of places, including:

1 Any DB2 family.

2 Oracle, Sybase, Informix, and SQL Server (IBM is not willing to give up any database market, particularly to Microsoft).

3 Flat files (two-dimensional array).

4 Data Joiner.

5 And, for very old timers who thought IMS and VSAM were dead and buried, the *Classic Connect Interface*!

IBM enhanced *Data Guide* renaming it *Information Catalog*. Its capabilities include:

1 Helping users find, understand, and access their data by providing:

- Data described in user business terms.
- User-friendly search engine.
- User-friendly communication between user and content owner.

- Access tool initiation.

2 Information sharing your way by providing:

- Support for any information object.
- Support for almost anything including databases, queries, Web pages, etc.
- Category grouping.
- Correct object metadata.
- Properties of almost anything.
- Support of almost any user authority.

3 Synchronization of information objects:

- Automatic population with *Data Warehouse Center.*
- Metadata interchange with most popular access tools including QMF, DB2 OLAP, Brio, Business Objects, Cognos, Hyperion, etc.

IBM made it easy to connect to IC by providing access to:

1 Windows (IBM does not want to, but has no choice) or a Web browser (IBM hopes for Navigator).

2 Almost any form of distribution including workgroups or a central repository.

3 Storage by any DB2 family.

4 Selected automatic population.

DB2WM provides additional functionality to let data warehouses better satisfy their objective of providing users with timely and accurate information for improving their bottom line. Additional UDB V7 performance and SQL enhancements allow DB2 star schema relational databases to be increasingly effective data warehouses (see *DB2 Update*, Issue 105, July 2001).

IBM's next major enhancement was Intelligent Miner for Data (IMD)

and Intelligent Miner Scoring (IM Scoring). Usually, the major data warehouse problem lies in providing useful information to users. It probably is the reason that IBM developed IMD as a DB2 extension for data mining. IBM's definition is, "The process of extracting *valid, useful, unknown,* and *comprehensive* information from data, and using it to make business decisions". The keyword is 'unknown'. OLAP works with known data (see *DB2 Update*: Issue 102, April 2001, and Issue 105, July 2001). and is verification-driven analysis, while data mining (DM) is discovery-driven, not needing human assistance or input.

DM techniques include:

- Association – a counting algorithm that determines the probability of multiple items occurring in a transaction by analysing past transactions. Often used for market basket analysis (7-11 presumably found that beer and diapers are often sold together on Monday nights during American football seasons.) Other uses include item placement planning and promotional sales planning.
- Sequential patterns – an association variation, where transactions of specific people are individually analysed instead of transactions in general. Often used by direct marketers to design specially-targeted advertising.
- Clustering – a segmentation procedure that divides a database into subsets (clusters) with each cluster member containing similar properties. Uses either demographics or IBM's Neural Network Utility (a family of tools incorporating fuzzy logic rules – see *DB2 Update*: Issue 87, January 2000; Issue 89, March 2000; and Issue 94, August 2000). Its uses include cross-marketing, cross-selling, and customizing marketing plans for different customer types.
- Classification – an automatic process for creating model classes from record sets based on their characteristics. Often used for developing specific promotional plans for identifiable customer classes.

There are more sophisticated techniques and frequently techniques are combined.

IMD is client/server whose clients can be AIX, OS/2, Windows NT, 95, or 2000. Clients can manipulate the data with visualization tools, or build a DM operation to run on the server with results returned for analysis. Servers can be OS/390, AIX, OS/400, Solaris, and Windows NT or 2000. Servers do the mining and processing functions storing the data and mining results. Mining can be performed directly on DB2, flat files, or any data source accessible by DataJoiner. A high-speed extract utility can import data into DB2 UDB from Oracle and Sybase. IMD can use parallel processing (see *DB2 update*, Issue 107, September 2001*)*. IMD also runs on Oracle8*i* Version 8.1.6 or later! International firms will be pleased that IMD is available in Brazilian, Chinese, English, French, German, Hungarian, Italian, Japanese, Korean, Russian, or Spanish.

IM Scoring workflow is:

- Data miners create model output as a PMML (Predictive Model Mark-up Language) file, forwarding it and the variable creation logic to IT.
- The database programmer or DBA stores in DB2 using BLOB for the model logic.
- SQL is used to execute the variable creation logic.
- The database programmer or DBA applies the modelling logic using IM Scoring functions to create a scored data set.
- The domain expert and data miner verifies output. Any debugging is done.

There are many advantages:

- The programmer or DBA can access model logic using a standard SQL API. This allows dynamic updating as business rules change.
- More complex algorithms can be used including factor analysis, linear regression, principal component analysis, univariate curve fitting regression, etc.
- IT can create SQL scripts that are managed and scheduled using standard DBA tools.

- Real-time and batch mode applications can be enabled by allowing them to access the database containing the model logic.
- Model variables do not have to be created in advance.
- SQL view can be used, so only model scores and record identifiers need outputting. Another view advantage is that it requires only a single database scan, allowing IT to manage its computing resources more efficiently.
- State-of-the-art GUI for visualization.

PMML is an eXtensible Mark-up Language (XML) used to provide a vendor-independent method of defining predictive models. The Data Mining Group (DMG) is PMML's sponsoring and governing body. Its Web site (http://www.dmg.org/) provides PMML syntax. DMG states, "…PMML is an XML-based language which provides a quick and easy way for companies to define predictive models and share models between compliant vendors' models. PMML provides applications a vendor-independent method of defining models so that proprietary issues and incompatibilities are no longer a barrier to exchange of applications between applications. …Previously (before PMML) this was virtually impossible."

I believe the most significant PMML technique, out of the dozen available, is Naïve Bayes Model. Reverend Thomas Bayes, whose vocation was mathematics, had his paper on *Bayes Theorem* published posthumously in 1763! It remained an unproven obscure statistical theorem, because his complex equations required computing power not available until the late 1980s. Bayes Theorem allows scientists to combine new data with their prior knowledge on how the universe works. Assume new-born babies who have never seen a sunrise. Their priority is fifty per cent for sunrise tomorrow and two-thirds for the day after. After fifty years, the probability of the sun rising is a near-certainty.

Several major organizations are now using Bayesian networks including:

- The US Navy, for identifying incoming enemy fire and recommending which weapons should be used for defence.

- Pfizer, for conducting 'humane' clinical drug trials. Bayesian methods allow them to alter the dosage during the trial, instead of giving control groups placebos.
- General Electric, for pinpointing emerging problems in their jet engines.
- NCR, for selecting which products or services to display on their ATMs, as users wait for their transactions to complete.

Bill Gates has established a Bayesian research team of twenty plus. Their first product was MS Office Assistant, which employs Bayesian methods to make suggestions to users based on their previous work. The next Bayesian product is rumoured to be a virtual secretary that will analyse incoming e-mails and phone calls to determine response priority.

I believe Bayesian networks will provide organizations with substantial improvement in their ability to make accurate predictions. Fuzzy SELECT (see *DB2 Update*, Issue 87, January 2000) lets users ask fuzzy questions such as, "Which customers have high income and live around Port Dickson?" Bayesian techniques will allow users to predict what those customers are likely to buy.

DATA WAREHOUSE SUMMARY

IBM's initial foray into data warehousing was supporting star and its variants in DB2, even though they violate Codd's normalization rules. They became serious when they included OLAP in DB2 UDB V6, allowing star schemas as an alternative. Another important enhancement was the addition of DB2 Warehouse Manager to DB2 OLAP, providing DBAs with simplified procedures and extensive connectability. IBM's renamed *Information Catalog* provides users with many options including metadata interchange with most popular access tools. IBM has had QMF for its query language from day one. It became obsolete when IBM released Intelligent Miner for Data with Intelligent Miner Scoring using PMML. IBM has made an all-out commitment to PMML, so it can become the leader in access languages.

I suspect that many readers wondered about my statement that data warehousing is the only practical way to leverage mainframe data. The following should convince sceptics:

1 A DBA designs a star or variant schema for the user requirements.

2 S/he uses simple tablespaces and other features to maximize performance.

3 The data can be retrieved from almost anything including any DB2 family, Informix, Oracle, SQL Server, Sybase, flat files, VSAM, IMS, and any other file available to DataJoiner.

4 The *Information Catalog* can be connected by Windows, any Web browser, workgroups, or central repository. Any DB2 family can store it and use automatic population.

5 Metadata interchange with Brio, Business Objects, Cognos, DB2 OLAP, Hyperion, QMF, and many more.

6 Advanced query techniques using Intelligent Miner for Data (IMD) and Intelligent Miner Scoring (DB2IMS).

7 IMD is client/server whose clients can be AIX, OS/2, Windows NT, 95, or 2000. Servers can be AIX, OS/390, OS/400, Solaris, or Windows NT or 2000.

8 DB2IMS uses PMML, which allows seamless interchange with any PMML model developed on any computer using any software.

9 PMML has sophisticated query techniques including statistics, taxonomies, and hierarchies, Tree Model, Naïve Bayes Model, General Regression Model, Association Rules Model, Neural Network Model, and Clustering Model.

WEB-ENABLING

The only practical way to Web-enable is WebSphere. It has three editions – Standard, Advanced, and Enterprise. This article deals only with Enterprise, because DB2 V7.1+ runs under OS/390 on

mainframes. Mainframes are used for heavy transaction volumes, which in turn require Enterprise. Enterprise provides:

- Dynamic Web content generation using Enterprise JavaBeans, XML, and XSL; and bean-managed and container managed persistence with EJB and container services.
- Reusable business logic and portable data for rapid site deployment.
- Usage analysers and reporting tools.
- Apache-based HTTP server among others.
- Database connection pooling for dynamic JDBC access to DB2, Oracle, and SQL Server.
- Tivoli-ready modules.
- XML parser utilizing the latest W#C XML 1.0 specifications.
- XSL parser for transforming XML data into formatted HTML.
- Lightweight Directory Access Protocol (LDAP).
- Common Object Request Broker Architecture (CORBA) and EJB.
- Control and protection.

Simple Web-based application architectures are 'three-tier' comprising a 'thin-client', where the Web browser executes; a 'middle-tier', where the Web server and application server operate; and a 'back-end' tier that contains the enterprise logic and database servers.

Mainframe Web applications usually employ distributed client/server applications ('*n*-tier') because they have a 'client-tier' that uses distributed objects encapsulating the business logic and 'back-end' enterprise data. Distributed objects can be clients of other distributed objects. '*n*-tier' typically requires richer server interaction. A standard technique is to use client Java applets that communicate to an application server with Internet Inter-ORB Protocol (IIOP) or Remote Method Interface (RMI) over IIOP.

Enterprise relational data can be accessed by a Java Data Base Connect (JDBC) API or SQLJ(ava) API. SQLJ is based on the ANSI X.3.135 SQLJ standard. It runs on top of DB2 JDBC drivers providing Java classes, translator (converts DB2 code to Java), and runtime/ customization tools.

DB2 JDBC driver supports type 2 (application driver support) and type 3 (applet/net driver). A JDBC requests flows to DB2 CLI to the DB2 Server via the DB2 Client Application Enabler (CAE). Tests indicate that type 2 provides better performance.

The data warehouse, or data mart subset(s), will probably provide most of the data, although some applications such as customer service call centres may use only the data warehouse. Transaction applications, such as e-business, require access to OLTP applications. Webmasters like to design Web pages that are hit so often that they say ouch! This illustrates a critical difference between data warehousing and transaction applications. Data warehouse queries are unpredictable and can require massive computer power to answer. Transaction systems follow a defined thread, even if there are many branches. A data warehouse cannot be directly used as input for a transaction application because it may provide unacceptable response time for transaction users such as surfers. Any Webmaster will tell you that a delay of a second can cause the surfer to surf somewhere else. So, rule one is that the data warehouse or data mart subset must be replicated for the Web page.

SUMMARY

The best way to leverage mainframe data is to collect it in a data warehouse. Cleansing and verification routines have vetted the data, assuring users it is correct and usable. WebSphere is the best way to Web-enhance using data warehouse historical data and OLTP current data. The data warehouse must be replicated to provide acceptable response time to the Web users.

References:

- *Servlet and JSP Programming with IBM WebSphere Studio and VisualAge for Java,* SG24-5755.
- *The Universal Connectivity Guide to DB2,* SG24-4894.

Web sites:

- WebSphere–http://www.ibm.com/software/webservers/appserv.
- Servlets – http://www.software.ibm.com/ebusiness/pm.html#Servlets/.
- JSP – http://www.software.ibm.com/ebusiness/pm.html#Java.Server.Pages/.
- JavaBeans – http://www.javasoft.com/beans/docs.
- Enterprise JB – http://www.java.sun.com/products/ejb/.
- Connectors – http://www.software.ibm.com/ebusiness/connectors.html/.
- XML – http://www.software.ibm.com/xml/.
- CORBA – http://www.omg.org/.
- RMI – http://www.java.sun.com/products/jdk/rmi/index.html/.
- IIOP – http://www.whatis.com/iiop.htm/.
- JNDI – http://www.java.sun.com/products/jndi/index.html/.
- JDBC – http://www.java.sun.com/products/jdbc/index.html/.
- JMS – http://www.java.sun.com/products/jms/index.html/.
- JTA – http://www.java.sun.com/products/jta/index.html/.
- SQLJ – http://www.sqlj.org/.
- DB2 Java – http://www.ibm.com/software/data/db2/java/.

Eric Garrigue Vesely
Principal/Analyst
Workbench Consulting (Malaysia)

HiT Software has launched its DB2Motion, which replicates iSeries and AS/400 application data to Microsoft SQL Server and Oracle application environments, providing near real-time mirroring of iSeries changes.

The DB2Motion Enterprise Manager runs on a Windows client and allows administration functions with wizards, enabling administrators to define destination SQL Server or Oracle tables.

For further information contact:
HiT Software, 4020 Moorpark Avenue, Suite 100, San Jose, CA 95117, USA.
Tel: (408) 345 4001.
URL: http://www.hitsw.com.

* * *

IBM has announced Version 1.2 of its Tivoli Data Protection for Enterprise Storage Server (ESS) Databases back-up recovery software, which now provides online back-up for DB2 and exploits the ESS disk subsystem with its new FlashCopy function.

The software, which is also available for Oracle databases, is said to virtually eliminate back-up-related performance impacts, downtime, and user disruption on the production host.

It's aimed at sites with expanding back-up windows or who need 24x7 application and database availability. Working with Tivoli Storage Manager, it promises consistent and reliable unattended online back-up and recovery of databases.

Key features include automated, high-performance backup through exploitation of the FlashCopy function of IBM's Enterprise Storage Server, virtually no back-up-related downtime of applications and databases, compliance and integration with Oracle and DB2 UDB databases, automated scheduling with Tivoli Storage Manager scheduler, and centralized control.

For further information contact your local IBM representative.
URL: http://www.tivoli.com/products.

* * *

JD Edwards has begun shipping its OneWorld Xe collaborative commerce applications on IBM's DB2 Universal Database for Unix and Microsoft Windows 2000 platforms. The new implementations promise exclusive DB2 optimization and configuration tools. Also, DB2 will now become JD Edwards' internal development and maintenance platform.

The ERP company reckons that, compared with other available databases, DB2 offers all of the features and functions of a leading database, and is less expensive to purchase, implement, and maintain. It is also easier to install, upgrade, configure, and tune.

It adds that it will support DB2 on a wide variety of platforms, including AIX, Sun Solaris, HP-UX, and Windows 2000 environments, as well as providing continued support for DB2 on iSeries running OS/400.

For further information contact:
JD Edwards, One Technology Way, Denver, CO 80237, USA.
Tel: (303) 334 4000.
URL: http://www.jdedwardsnews.com.