

IBM DB2 Version 7 Utilities Cool Features and Hot Performance!

Klaas Brant – KBCE
Susan Lawson – YL&A, Inc

Every DB2 release is packed with lots of new features. So every new release gets a lot of attention. From the moment that Version 7 was announced the DB2-L list server was overloaded with messages about the changes to the utilities. What happened? IBM announced that they were going to charge for their utilities. From the moment this news became public it was one of the most discussed issues in the DB2 market in history. But all discussions were without conclusion because nobody actually knew what the packaging or pricing would be. Those mysteries have been uncovered and now it is time to get back to the technical details and determine how to make informed decisions about our utility choices.

Brief History of New Offerings

This paper is a technical paper and not about pricing or politics. However it does help to understand the background of the issues with pricing so informed decisions can be made regarding utility selection.

In the late 80's and early 90's it became clear that the S/390 platform was becoming expensive relative to smaller platforms. Windows/Intel and Unix began to attempt to offer the same performance as the mainframe but for a lower price. And this has become a reality; a desktop PC today with a 1 GHz processor has the same power as these older mainframes. However this still cannot beat many of the known advantages of the mainframe technology such as high availability, high concurrency, robust change control and a reliable backup and recovery to name a few.

In order for this technology to continue to thrive it was important to lower the total cost of ownership of the mainframe. This was done on the hardware side (e.g. with the cheaper CMOS processors) and on the software side with better pricing structures. IBM did make successful changes in these areas and the market for S/390 continues to be stable and growing. However, not all vendors on this platform were willing to change pricing structures and this became a downside for customers who wished to upgrade their mainframe technologies. For many, the cost to maintain DB2 became more expensive than the cost to own it.

In order to keep cost down, IBM now offers alternatives for expensive third party software. This software market for DB2 is concentrated in tools and utilities. In recent releases of DB2, IBM has begun to offer additional utilities and enhanced functionality. Version 6 has provided many

enhancements to utilities in the areas of productivity and performance, making them comparable to third party utilities. Now with Version 7 and IBM's new competitive initiative, the utilities are being greatly enhanced and there are also several additional utilities being offered. There will always be unique features and functionality in the third party tools, but the gap in performance is quickly closing.

Packaging the utilities

After you have installed DB2 Version 7 the utilities will function for a period of 30 days. After this period the utilities will have to be enabled as being separate products, although the utilities will ALWAYS function on the DB2 catalog (DSNDBxx). As always, it is a good idea to use the IBM utilities on the DB2 catalog because the utilities often use special code for these objects. If you purchase the IBM utilities you can buy the following suites:

- **DB2 Operational Utilities**
 - COPY
 - EXEC SQL
 - LOAD
 - REBUILD INDEX
 - RECOVER
 - REORG INDEX
 - REORG TABLESPACE
 - RUNSTATS
 - STOSPACE
 - UNLOAD

- **Recovery and Diagnostics Utilities**
 - CHECK DATA
 - CHECK INDEX
 - CHECK LOB
 - COPY
 - COPYTOCOPY
 - MERGECOPY
 - MODIFY RECOVERY
 - MODIFY STATISTICS
 - REBUILD INDEX
 - RECOVER

- **Full Utilities Suite**
 - CHECK DATA
 - CHECK INDEX
 - CHECK LOB
 - COPY
 - COPYTOCOPY
 - EXEC SQL

- LOAD
- MERGECOPY
- MODIFY RECOVERY
- MODIFY STATISTICS
- REBUILD INDEX
- RECOVER
- REORG INDEX
- REORG TABLESPACE
- RUNSTATS
- STOSPACE
- UNLOAD

The base DB2 licensed product has the following utilities:

- CATMAINT
- DIAGNOSE
- LISTDEF
- OPTIONS
- QUIESCE
- REPAIR
- REPORT
- TEMPLATE
- DSNUTILS (utilities stored proc)
- DSN1 (standalone utilities)
- DSNJ (BSDS and log maintenance)

The basic idea is that everything that has an “equivalent” by third party vendors is removed from the base DB2 licensed product. There is some overlap between the utility suites to allow people to buy only one suite but the MODIFY utility will make the Recovery and Diagnostic suite a must have.

Remember: you MUST use the MODIFY utility or you can experience performance degradation.

For the same reasons the REORG utility in the Operational Suite is a must have, unless you have a data warehouse environment but then LOAD is a must.

Note: The Full Utilities Suite is delivered with the product and enabled for use on the System Catalog Tables without additional cost.

New Version 7 Utility Features

All utilities have been enhanced and there is even a new utility: UNLOAD. By upgrading the utilities with wildcard capabilities and dynamic allocation the utilities have become state of the art utilities that can match up with

any other utility on the market. This paper will take a look at the details to all enhancements to the utilities.

First, we will take a look two very powerful features that are embedded in all utilities: Dynamic Allocation and List Processing for objects.

Dynamic Allocation

Once you have experimented with this new feature you will find out that this is a very powerful feature. By means of “templates” you can get full control over the datasets allocated by the utilities. Together with the wildcard options you now have the power to build a utility job that you don’t have to update all the time if you have changes to the object structure. This is a quantum leap forward in terms of usability. Also environments that change all the time, like the SAP/R3 environment, will greatly benefit from these dynamic features. In order to use it you have to create one or more templates. A template is identified with a TEMPLATE statement and has 1-8 char names. Where in previous releases you referred to DD cards in the control statements you can now refer to a template. The TEMPLATE statements can come from:

- TEMPLATE DD statement, which is processed before SYSIN
- SYSIN DD statement where they will precede the utility statement(s)

Any statements in SYSIN with the same name as a template from the TEMPLATE DD statement will overrule one from TEMPLATE DD statement. This gives you the ability to code a corporate standard for templates that can be generic for ALL jobs. If needed you can overrule the corporate standard in the SYSIN. Now what if the template refers to a name that also exists as a DDNAME in the JCL? In this case the template is ignored and the DD statement will overrule the template. If you combine dynamic allocations with wildcards then it becomes almost impossible to overrule the templates in the JCL so to stay flexible do not use this feature. In a template you can specify the parameters for:

- Dataset naming conventions
- DF/SMS parameters
- Special DASD or TAPE parameters
- Many other things you seldom need because the utility will set these parameters correctly for you (e.g. space parameters)

The utility driver will calculate the space needed for the datasets (see also keeping your RUNSTATS current). You can control if datasets go to DASD or TAPE depending on calculated size. This

is a very cool feature. If you want to use GDG's then you can ask the utility driver to define the GDG base if it does not exist yet. When you use tape datasets then you have the option to stack datasets on the tape. All these features are restart proof... try this yourself with stacked datasets on tape!

Coding the Template

Templates are expanded at execution time and have to come up with a dataset name. You can control the dataset name using variables and literals, very much like you do in JCL procedures. Next to the object names many other environment variables are available as variables to build the dataset name. You can use variables like job name, subsystem name, utility name, list name, and sequence number of object in the list to build a unique dataset name. A complete list can be found in the utility manual. Also date and time variables are available in many formats. The date and time are set only once when the utility starts and remembered by the utility in SYSUTLX. So when a utility is restarted the "old" date and time variables will be used to prevent all kinds of operational problems. Sometimes a variable can only be used with a certain utility, e.g. &ICTYPE is only valid when used in a COPY utility. Sometimes variables take each other's place e.g. when &TS (table space name) is used in a COPY of an index the index space name is substituted. If you use the &PART on non-partitioned objects, it will substitute 0000. This is all done to allow you to code simple and straightforward templates.

Example 1:

```
TEMPLATE KBCEIC1
DSN(&DB..&TS..&PRIBAC.&PART..D&JDATE)
COPY TABLESPACE DBKB01.TS001
COPYDDN(KBCEIC1,KBCEIC1)
```

Even in this simple example you can see the power. The template is used twice in a single COPY statement. The variable &PRIBAC will hold "P" for the primary image copy and "B" for the backup image copy. So here the utility will allocate 2 dataset names:

- DBKB01.TS001.P0000.D01120 for the primary dataset

- DBKB01.TS001.P0000.D01120 for the backup dataset

Allocation is done by ACS routines like any other dataset you specify in JCL. The sizing is done by the utility. If used in combination with SMS guaranteed space there will no longer be x37 abends anymore.

Example 2:

```

TEMPLATE KBCEIC1
DSN(&DB..&TS..&PRIBAC.&PART..D&JDATE)
      MAXPRIME(250) NBRSECND(5)
DASDMAX(1000,KBCETA01)
TEMPLATE KBCETA1 DSN(&DB..&TS..&PRIBAC.&PART(+1))
      MODELDCB(KBCE01.MODEL.COPY)
GDGLIMIT(10) UNIT(CART)
      STACK(YES)
COPY TABLESPACE DBKB01.TS001 COPYDDN(KBCEIC1)

```

Here is a more complex example. The KBCEIC1 template tells the utility never to allocate a larger quantity than 250 MB and to allocate the secondary extents so that there will be 5 extents and that if the dataset goes over 1000 cylinders it should use the KBCETA1 template. The KBCETA1 template creates stacked tape files. The datasets are allocated using the GDG technique. If the GDG base does not yet exist then it invokes IDCAMS to define one with a limit of 10 generations and a model dataset. The sizing trick can also be done the reverse way. You specify a tape template and with the TAPEMIN keyword you can divert datasets to DASD if they are small. If you think this is powerful then wait until we combine this with a wildcard for objects!

Keep RUNSTATS Current

The sizing for the datasets is done automatically based on RUNSTATS figures and information from the ICF catalog. So you have to keep the RUNSTATS figures up to date. Of course it is good practice for other reasons to maintain the stats. For COPYDDN and RECOVERYDDN the utility will use the HI-USED RBA from the ICF catalog. This is another cool feature because this figure will always be correct. Suppose you have an empty object e.g. a LOAD of a new table. In these cases it is still your job to specify the correct space parameters (except for COPYDDN and

RECOVERYDDN). You can do this in the template, but it might be just as easy to do it in normal JCL. The utility will always try to allocate the datasets without extents. So it calculates a primary quantity to hold the dataset and uses 10% of this primary quantity to allocate a secondary quantity. Again you can overrule this as shown in the previous examples.

Dataset behavior

All datasets are allocated with the correct disposition to allow for utility restart. There is nothing magical about dynamically allocated dataset. The limit is still 44 characters with a maximum of 5 levels, so be careful: you can easily create a dataset within a template with an invalid name. The datasets behave like datasets in JCL. For example, if you code a dataset that is being used in another job then you will be enqueued and have to wait for this resource. The utilities only do allocations and do not do deallocations. The MVS initiator does deallocation. This means that datasets will be cataloged at the end of the job step (or job in case of GDG's) like in normal JCL processing. Templates only work for output datasets and not for input datasets. So you cannot allocate your load dataset with a template. The sort also needs extra datasets like SORTWKxx. If SORT is configured correctly, SORT can dynamically allocate these datasets.

LIST processing for objects

Although many people refer to the new feature as wildcards, the official name is LIST processing. The utility will build a LIST of objects to process based on your LISTDEF statement. In this LISTDEF statement you can include and exclude objects not only based on wildcards that will be used in pattern matching with the objects, but also on RI related, Partition related, LOB related and/or non-LOB related objects. You can include other lists or select objects with a certain attribute (e.g. indexes that can be copied). The whole thing can become quite complex but there is a good testing facility available. My advice is KISS (Keep It Simple Stupid!). There are a few things to remember before you start building your object lists:

- The utility will create a list in the UTILINIT phase and store this list in SYSUTILX.
 - The LIST is static after the UTILINIT phase. This can become a problem in very dynamic environments. An object further in the list can be unavailable or dropped once the

utility selects this object. This can also be a problem if there is time between a failure and restart. During restart the original list from SYSUTILX will be used. (see also error processing)

- A list cannot be empty.
- If after all the filtering the list becomes an empty list the utility will not accept it any more and will fail.
- Duplicate objects are removed from the list.
- A list has NO order (not even after removing duplicates).
- Catalog objects (if included in the list) will be removed from the list.
 - List processing CANNOT select catalog objects. You have to give fully specified object names if you run utilities against the catalog (so no DSNDB06.* processing).
 - This is because utilities against a catalog object often have a sequence (e.g. the order to COPY the catalog objects).
- A list can select both table spaces AND index spaces but many utilities only accept one type of object.
 - If the utility does not accept a selected object then an error will be generated (see Testing And Dealing With Errors)

When you build a list it is referred to with a 1-8 character name. You use this name in combination with the keyword LIST in the utilities. All utilities except CHECK support this option. The -DIS UTILITY command has been enhanced so it now shows how many objects there are in the list and what object is being processed at the moment.

Let us review some examples to explore the LISTDEF statement:

Example 3:

```
LISTDEF KBCOPY INCLUDE TABLESPACE DBKB01.*  
PARTLEVEL  
TEMPLATE KBCEIC1  
DSN(&DB.&TS.&PRIBAC.&PART..D&JDATE)  
COPY TABLESPACE LIST KBCOPY COPYDDN(KBCEIC1)
```

Here we ask for a list containing all table spaces in the database DBKB01. We want the partition granularity. In case we select non-partitioned table spaces also, then

PARTLEVEL is simply ignored and the template will use 0000 as discussed before.

Here is another powerful example:

Example 4:

```
LISTDEF KBRI INCLUDE TABLESPACE DBKB01.TSKB01 RI
TEMPLATE KBCEIC1
DSN(&DB..&TS..&PRIBAC.&PART..D&JDATE)
COPY TABLESPACE LIST KBCOPY COPYDDN(KBCEIC1)
```

Here we ask for a list that contains the table space DBKB01.TSKB01 and all other table spaces that are connected using DB2 enforced RI. This means that the TABLESPACESET option that is available for REPORT and QUIESCE is now available for almost all utilities.

In the examples it becomes clear that if you use the LISTDEF you also have to use the TEMPLATE statements. This is because the utility will now have a variable list of objects to process and you have no option to pass a DDNAME for every object.

Guidelines for LISTDEF and TEMPLATE

As always there are some pitfalls for new options. There are a number of things you can and cannot do. Here are things you have to think about or have to avoid:

- Utilities have their internal restrictions.
 - These restrictions still apply in V7. For example, there is a limit to the number of table spaces you can QUIESCE. Although this number was raised several times in past releases, the number is now not fixed and depends on resources available. A QUIESCE with a LISTDEF TABLESPACE *. * will probably NOT work unless you have a small environment.
- Utilities start to use more and more resources if you give them a huge number of objects to process.
 - This can backfire on you and the utility slows down because of the memory usage and other resources used (e.g. SYSUTILX processing). Be reasonable in your requests.
- From an MVS point of view nothing has changed.
 - The utility is a batch job that uses memory and CPU and is allocating datasets. There is a limit to the number of datasets you can allocate in a single job step and for the

total job. There is a risk that with LISTDEF/TEMPLATE processing you have to make sure that you don't hit those limits. By crossing these limits you risk strange abends without debug information (e.g. batch job terminated at end of memory).

- JES3 is different and has restrictions on dynamic allocation for tape units.
 - Carefully read the manuals before you start to code or test.
- DSNUTILS, the stored procedure to invoke utilities, has been changed to support the new options.
 - You will have to change the calling program to use the new options otherwise DSNUTILS will still use it's own allocation.
- RI and PARTLEVEL are mutually exclusive.
 - Even if you specify PARTLEVEL without reference to a partition (so the whole table space) the RI keyword cannot be used.

Testing and dealing with errors

These new options are so powerful that you need to test to see if everything is coded correctly and if the results are what you expect them to be. The people who build these new facilities made excellent testing facilities for us. Just as TYPRUN=SCAN in JCL does not run the job the utility driver does not invoke the utilities if you code a restart PARM of PREVIEW (either UTPROC in DSNUPROC or the third parameter if you invoke DSNUTILB yourself). This special restart parameter will expand all LISTDEF and TEMPLATE statements so you can check if the utility is doing what you want it to do. Warning: this does not mean that if PREVIEW ends with RC 0 that your utility statements are OK. For example, you can pass a table space to a REBUILD utility. Because the real utility is NOT invoked this error will go unnoticed to the PREVIEW facility. PREVIEW can also be specified in the new OPTIONS statement.

How does a utility deal with objects it is not able to process (e.g. the object is in RECP status)? By default the utility will halt on these errors but you can overrule the default action. If you code OPTIONS EVENT(ITEMERROR, SKIP) then the utility will generate messages and skip the object. This option does NOT trap abends during the processing of an object. Another keyword you can specify on the OPTIONS statement is how the utility driver should set the return code if warning messages were generated. If you code OPTIONS

EVENT(WARNING,RC0) then the utility will end with return code 0 instead of 4.

Now let us explore new features of the utilities.

UNLOAD

Several people do unloads of their data from DB2 tables to process the data in sequential batch streams or pass them on to other systems. The most common technique to do this in the past has been to use the unload program DSNTIAUL. The problem with DSNTIAUL is not only that it is not very rich in function but also it is also very slow. In fact it has the speed of SQL, because it is SQL. Utilities are typically much faster because they attack the table space at a low level.

In Version 6 a feature was introduced in the REORG utility to unload data. This was a big improvement but still did not have many features. Version 7 comes with a complete new utility: UNLOAD. It has many features including the ability to unload from image copy. It also can make use of parallelism when unloading a partitioned table space in which case it outperforms REORG. With this new utility you have total control over which columns you want to unload and in what format you want to unload including conversions to ASCII or Unicode using CCSID's. You can also ask for only a sample of the data instead of a full unload, but when unloading samples from several tables you will NOT be able to create a set of files where the RI is consistent.

The output records written by the UNLOAD utility are compatible as input to the LOAD utility. This provides the ability to reload the original table or different tables with the data from the UNLOAD.

Isolation level

You can specify the isolation level using the SHRLEVEL keyword. Like in other utilities you have the option for REFERENCE or CHANGE. The REFERENCE options will drain the SQL and ask for exclusive control of the data. The CHANGE option has two sub options (only available in UNLOAD): ISOLATION CS and ISOLATION UR. These sub options are the same as in SQL whereby the utility will assume CURRENTDATA(NO) for the ISOLATION CS

What cannot be unloaded

To unload data a correct description of the table should exist in DSNDB06. The UNLOAD utility accesses the table spaces or image copy directly bypassing SQL processing.

The following cannot be unloaded:

- Dropped tables
 - They cannot be unloaded from an image copy that still exists. Even if you recreate the table, you still will be unable to unload the data (the OBID number will have changed when you recreate)
- DSNDB01 cannot be unloaded
 - Some of the objects in DSNDB01 are not even real tables and also DSNDB01 does not have entries in DSNDB06.
- LOB data > 32K
 - A MVS restriction is that records cannot be larger than 32K. LOBs can be (much) larger than 32K. To unload LOBs you have to unload the base table.
- Global temp table
 - These tables are empty until SQL processing puts data into them
- Views or SQL statements as input
 - This is a powerful feature of DSNTIAUL but requires SQL processing
 -

Input and output

The UNLOAD utility can unload from both table spaces and image copies. The image copy cannot be a concurrent copy (this is a backup, not an DB2 image copy). All other image copy formats like incremental copy or online copy are allowed. The UNLOAD can give warnings in non-full image copies (e.g. duplicate data could exist in inline copies). If you need a consistent full set of data you should use a FULL SHRLEVEL REFERENCE image copy.

Data can be unloaded into a single dataset or multiple datasets. If you unload multiple tables into a single dataset then the records will be tagged with an identifier to mark the table. The (optionally) generated LOAD statement will use these tag identifiers to load into multiple tables again. If you unload a partitioned table space and allocate a dataset per partition using templates then the utility will go into multi-tasking speeding up the process using IO overlap.

An example of the syntax for the UNLOAD utility is shown below.

```
UNLOAD TABLESPACE DB2DB.DB2TS
FROM TABLE DB2USER1.EMPLOYEE
WHEN (EID = 300 AND SALARY > 90000)
```

RUNSTATS

We all know how important it is to maintain the statistics in the catalog. Not only does the optimizer use these figures, but also the DBA uses these figures to alter objects or run utilities when certain thresholds have been reached. In Version 7 there will be more statistics gathered. An important one is the number of extents on disk, so you now have a single source to make DBA decisions.

For a DBA not only are the current figures important, but also a history of these figures is important. Using a history you can see growth and pinpoint changes in the statistics to changes in the environment. Third party vendors not only created their own version of RUNSTATS but also provided the DBA with a history. In Version 7 the RUNSTATS utility can also collect a history. This option cannot be enforced and the default is NOT to collect a history so you have to change your RUNSTATS statements in order to use this feature

The REORG and REBUILD utilities, which can also run RUNSTATS in parallel, have been changed as well to allow the new HISTORY keyword.

```
REORG INDEX ..HISTORY
RUNSTATS TABLESPACE ..HISTORY
```

If you collect RUNSTATS with the new HISTORY keyword then the new statistics are also written in the _HIST tables. There is also an option to choose NOT to update the normal tables used by the optimizer (UPDATE NONE) but still write a history (HISTORY ALL). Using this great feature you can query the history tables and create your own trend analysis reports using a report writer.

The historical statistics are kept in nine different DB2 catalog tables:

- 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

MODIFY

The MODIFY utility has been enhanced to allow you to purge old RUNSTATS history. Like the SYSCOPY table the new _HIST tables can also grow quite large and you need a mechanism to delete records that were written before a specified date or that are older than a specified number of days. The modify utility now has in addition to MODIFY RECOVERY a MODIFY STATISTICS. The options are straight forward (delete by age or cutoff date) and there is no impact on data if you run or terminate the utility.

LOAD

An increasing number of companies are exploring the power of the S/390 or zSeries platform to house very large data warehouses. IBM understands there are issues with getting the required amounts of data into these large warehouses in a timely manner and is extending the LOAD utility with many new features to help speed up the process.

One problem in the past has been the time it takes to load massive amounts of data in a short window. Many users had to create multiple jobs that ran in parallel in order to stay within a given window. In this scenario, every job loaded an individual partition, but because the access to the Non-Partitioning Indexes (NPI) was not serialized there was a lot of contention on the NPIs. This slowed down the individual batch jobs. Some people even dropped their NPIs during LOAD and recreated them afterwards. With the new parallelism this problem should not occur any more.

LOAD parallelism

Version 7 combines the multi-tasking normally done by multiple jobs back into one job and creates one task to build the indexes per partition and the NPI's. This is a single sort and there will be no contention. Also, the Parallel Index Build (PIB) introduced in Version 5 now supports the new LOAD parallelism. PIB is activated by the SORTKEYS keyword and creates a sort task per index (both PI and NPI). The multiple REBUILD tasks (one per part) interface with all the SORTBLD tasks.

The number of parallel load tasks will be determined by the number of CPUs and virtual storage available, and the number of threads available. The following is an example of the necessary syntax:

```
LOAD INTO TABLE tab1 PART 1 INDDN infile1
      INTO TABLE tab1 PART 2 INDDN infile2
```

SHRLEVEL CHANGE

Sometimes the availability of the data is an absolute must. However, there are times when there is a need to load data into a table. In the past, the options were LOAD RESUME YES, which of course makes the table data unavailable, or to write a user program to perform inserts. The second option takes a lot of time (code, bind etc.) and is not a user-friendly solution. However there are some other advantages to using SQL INSERTs instead of the LOAD utility. For example, INSERT Triggers will fire, something the LOAD does not do. DB2 also optimizes the insert process by creating and using an I-PROC (kind of small program that does the insert) for reuse by the next INSERT.

The solution to this problem arrives in Version 7 with the new SHRLEVEL CHANGE, which is often referred to 'Online LOAD Resume'. With this new SHRLEVEL the LOAD will do normal SQL INSERTs. It does normal CLAIM processing and no DRAIN as a normal utility does. All related activities occur as they would with SQL: RI, Triggers, Duplicate Keys, Free Space processing etc. This is YGWYAF (You Get What You Ask For) processing. The BIG bonus: 100% availability and easy to use (no need to code a program). The only things that are different from SQL processing are that you need LOAD privilege (not INSERT privilege) and that the utility uses a utility timeout instead of an SQL timeout. Of course the LOAD does an optimized load by interfacing directly to the Data Manager so no SQL PREPARE or authorization check is needed. Commit processing is also handled by the utility. LOAD will constantly monitor the locking situation and adjust the commits based on the situation. Restarting is allowed since the utility does take checkpoints.

Once we are doing SQL INSERT processing, what is stopping us from doing an INSERT into a table in another server using DRDA? Nothing, since IBM has delivered this and the end result is a cross-loader technique that allows you to load data over a network.

REORG

There have been many changes to the REORG utility. The online REORG was a major improvement but there are still a few things to improve. The time needed in the switch phase is too high. An IDCAMS RENAME takes 1-3 seconds in a well-tuned system, so if you rename 255 parts and 255 PI's you wait for minutes even with some optimizations done inside DB2. So we have to get rid of the

RENAME scenario and this is the Fast SWITCH. Also the BUILD2 phase that updates the NPIs is taking too long. Version 7 has enhanced this phase to do things in parallel. The last bottleneck is the DRAIN that locks out Application processes. Version 7 will give you more control over the DRAIN processing.

Fast SWITCH

The normal scenario is that the online REORG will build a shadow dataset (S-datasets). Once the utility is ready to switch, it will rename the I-datasets to T-dataset and the S-datasets to I-datasets. If this is successful then the T-datasets can be deleted. In total, 3 actions for IDCAMS were required. The new option does not do this any more. With fast SWITCH you create a new J-dataset instead of an S-dataset. Once the system is ready to switch it will update DB2 control blocks and catalog to indicate that the J-datasets are now the current datasets. The I-datasets are deleted afterwards. The next time you do a REORG the same things is done but then with J-datasets becoming I-datasets again. So you alternate between I-datasets and J-datasets. This feature only uses IDCAMS to delete datasets, which is something the utility doesn't have to wait for. You can update the ZPARM (SPRMURNM=1) to indicate that ALL online REORGs should use this feature. The recovery utility has been updated to deal with this new feature and SYSCOPY will contain information about the datasets used.

BUILD2 Parallelism

During the BUILD2 phase the NPIs are locked out for SQL processing. Version 7 will attempt to do the BUILD2 processing in parallel. You have no control over this in the utility statement. The maximum can be controlled by the SYSPROG in the ZPARM SRPRMMUP. DB2 will evaluate calculate the number at runtime based on number of processors, available DB2 threads and number of logical partitions.

The DISPLAY UTILITY command has been enhanced to show the number of records updated for all logical parts. There is a new DSNU message that can be viewed with a DISPLAY command when issued during the BUILD2 phase that will indicate the number of logical partitions updated in parallel.

```
DSNU1114I      LOGICAL PARTITIONS WILL BE LOADED IN  
PARALLEL,
```

DRAIN processing

Utilities force themselves into SQL processing using a DRAIN. A DRAIN has a timeout, which you specify on the DSNTIPI panel (IRLMRWT * UTIMOUT). When a utility is unable to DRAIN the resources needed because an SQL process does not commit a number of other SQL processes, it will be timed out due to the DRAIN in control. What you want to have is a shortened DRAIN timeout for the online REORG with a retry option. This is exactly what the new DRAIN_WAIT, RETRY and RETRY_DELAY do for you

Conclusion

The new DB2 Version 7 Utilities are state of the art and are becoming very competitive with all other utilities on the market. Version 7 brings new features and functionality into the utilities as well as many performance improvements.

Klaas Brant, KBCE Netherlands, and Susan Lawson, YL&A, are both IBM DB2 and S/390 Gold Consultants. They both have several years experience in database and systems tuning. They also are the co-founders of the DB2 Symposiums, worldwide DB2 educational events. Klaas can be reached a kbrant@kbce.nl, and Susan can be reached at Susan_Lawson@ylassoc.com.