Spring 2009

THE BUFFER POOL

Utility Improvements in DB2 9 for z/OS

By Craig S. Mullins

Every new release of DB2 brings with it new functionality and improvements for the IBM DB2 utilities. And DB2 Version 9 is no exception. So let's take a high level look at the many enhancements made to the DB2 utilities in Version 9.

TEMPLATE Switching

One nice new feature that impacts multiple utilities is TEMPLATE switching. For those not familiar with templating, TEMPLATE is a utility control statement that enables the allocation of data sets for a LISTDEF (LISTDEF is the way you control the list of database objects for a utility execution). With the TEMPLATE statement you essentially specify data set naming conventions and allocation information without using JCL DD statements.

OK, so what is TEMPLATE switching? What it enables you to do is to specify different characteristics for image copies of varying sizes. So, for example, you can create a different template for low and high volume image copies. Here is an example:

```
//SYSIN DD *
TEMPLATE low DSN &DB..&TS..IC.D&DA..T&TI.
  UNIT=DASD LIMIT(50 CYL,high)
TEMPLATE high DSN &DB..&TS..IC.D&DA..T&TI.
   UNIT=TAPE
COPY TABLESPACE MY.SMALLTS COPYDDN(low)
COPY TABLESPACE MY.LARGETS COPYDDN(low)
```

Note the new parameter named LIMIT. This is what controls when the TEMPLATE is switched. In this example we set the LIMIT for low volume at 50 cylinders. When this is reached the TEMPLATE will be switched to the high volume template. Of course, you

have the flexibility to control the limit by specifying whatever number is appropriate for your site as well as specifying it in CYL, GB, or MB.

DB2 can only switch the TEMPLATE once, so you cannot have more than two templates. Template switching is available for image copies produced by COPY, COPYTOCOPY, MERGECOPY, LOAD, and REORG. And you can set up template switching for both the COPYDDN and RECOVERYDDN.

More Online Utility Capability

Under DB2 Version 9, three more utilities can run online – that is, with SHRLEVEL CHANGE. First up we have **CHECK DATA**. This utility is executed when you want to make sure that the data in your tables matches your integrity constraints. Prior to Version 9, access to tables (or table spaces) is read only when CHECK DATA is being run. This means that as long as CHECK DATA is executing you cannot modify data. If you have complex referential sets and/or very large databases you likely will have issues with running CHECK DATA.

So, once you are up and running with Version 9 you can run the CHECK DATA online; that means with SHRLEVEL CHANGE. As with other online utilities, an online CHECK DATA will work with on a shadow copy of the data and indexes. DB2 makes the shadow copy using DFSMS ADRDSSU. This copy can be super fast if you have data set level FlashCopy V2 enabled. Even if you do not have FlashCopy availability will improve because the outage is reduced when you use SHRLEVEL CHANGE (just to make the copy) instead of SHRLEVEL REFERENCE (for the entire utility execution).

DB2 will check data integrity using the shadow copies of the data and indexes. When violations are found DB2 will not set the CHECK pending state. And remember, we are checking shadows, so DB2 will not delete rows in violation. Instead, CHECK DATA will create a PUNCHDDN containing REPAIR LOCATE DELETE input that can be run against the active data.

The **CHECK LOB** utility has also been enhanced to enable online execution. Version 9 allows SHRLEVEL CHANGE for checking LOBs using a shadow copy of the LOB table space. And just like CHECK DATA, DB2 uses the DFSMS ADRDSSU utility to create the shadow. And similarly, the check is executed against the shadow and REPAIR statements are generated that you subsequently can run against the actual LOB table space.

Finally, you can also run the **REPAIR LOCATE** utility specifying SHRLEVEL CHANGE. This enhancement allows us to execute the REPAIR utility online with LOCATE against indexes, index spaces and table spaces.

COPY Improvements

IBM has also made several nice improvements to the COPY utility. First up is a new parameter that gives you the ability to copy just the pending database objects. The new parameter is called SCOPE PENDING, and when you specify it the COPY utility will only copy objects in copy pending or informational copy pending state. Of course, the default is SCOPE ALL, which makes COPY act like it always used to act; that is, without regard to the pending state. This option can be used in conjunction with LISTDEF to build very powerful, selective backup jobs.

The IBM COPY utility also has been extended to support copying clone tables (which are also new in DB2 V9). The new CLONE parameter must be coded on the COPY specification in order for clone table and index data to be copied. It is important to understand that the CLONE parameter is basically an on/off toggle. If CLONE is coded, then COPY will copy only clone table (and/or index) data. You can specify both clone and regular database objects in the same COPY specification, but the IBM COPY utility will ignore non-clone objects when the CLONE parameter is specified; and it will ignore clone objects if the CLONE parameter is not specified.

DB2 Version 9 also improves the functionality of the CHECKPAGE parameter. Recall that specifying CHECKPAGE tells the COPY utility to check each page in the table space or index for validity. If it finds an error, COPY issues a message that describes the type of error. Using CHECKPAGE is a good idea because it will identify problems in your page sets.

In previous versions of DB2, using CHECKPAGE with COPY was problematic because of high overhead and availability issues. In DB2 Version 9 IBM reports that the CPU overhead is about 5 percent for COPY INDEX and 14 percent for COPY TABLESPACE if you are running V8. As for the availability issue, as soon as a validity problem is found on a page, the COPY utility will place the page set into COPY PENDING so it cannot be accessed while COPY chugs merrily along looking for additional invalid pages.

Both problems are addressed in DB2 Version 9: IBM has significantly reduced the CPU overhead and fixed the availability issue. Instead of putting the page set into COPY PENDING immediately when it finds a problem, instead COPY will produce the message that it found a problem, but it will not set the pending state. Furthermore, it will continue checking pages but not copying anything.

Upon completion, COPY will issue a return code of 8 and it will update the SYSIBM.SYSCOPY catalog table indicating it found a broken page. This information is recording in a new column, TTYPE, with a value of “B”). This is important to understand because the page set will not be in a COPY PENDING state so you will have to check for the return code and fix the problem.

RECOVER Improvements

IBM has also made some nice enhancements to the RECOVER utility in Version 9. The most significant enhancement centers on recovering to a point-in-time (PIT) with

consistency. As any DBA who has ever been charged with recovering an operational database to a prior PIT knows, it can be challenging to accomplish. Of course, if you have a usable QUIESCE point then point in time recovery is easy. But that means you'd have to have planned ahead of time and taken a QUIESCE, which is not always possible with heavy 24x7 workloads.

Well, DB2 V9 offers some relief because the RECOVER utility is enhanced to automatically identify uncommitted work at the PIT to which you are recovering. After detecting the uncommitted work DB2 will rollback any changes made to the database objects that are being recovered. This means that anything that is being recovered will be in a transactionally consistent state. Of course, you still have to make sure that all of the appropriate objects are participating in the recovery.

This works with for PIT recoveries executed with either the TOLOGPOINT or TORBA keywords only. To accomplish this two new phases are added after the LOGAPPLY phase: LOGSCR and LOGUNDO.

The LOGSCR phase runs for each DB2 member with an active unit of recovery (UR). If no UR was active then this phase will be skipped. During the LOGCSR phase, all active and incomplete URs are identified. This includes URs that were INFLIGHT, INABORT, INDOUBT, and POSTPONED ABORT.

If any URs were found, the LOGUNDO phase runs. This is where RECOVER backs out the changes made on recovered objects by active URs. In a data sharing environment, this will be done one member at a time.

Another improvement made to the IBM RECOVER utility in V9 is the ability to gather more information about the progress of the recovery. Any DBA who has been tasked with sitting around and monitoring recoveries what the DISPLAY command to show accurate, up-to-date, and appropriate information about the status of the recovery.

Version 9 enhances the DISPLAY UTILITY command to provide additional information during the LOGAPPLY phase. DISPLAY now will show the range of the log that needs to be applied, for the database objects that are being recovered. It will also show the progress up to the last commit, as well as the elapsed time since the start of the log apply phase of the recovery. This information will assist the DBA in determining how long it will take for the recovery job to successfully complete.

One final enhancement made to the RECOVER utility in V9 gives the DBA control to get DB2 to recover using an earlier image copy. This is accomplished by specifying a point in the log before which DB2 must select an image copy for the recovery.

This technique can be particularly helpful when the most recent image copy is damaged or somehow unusable. You can use an earlier copy by providing an RBA/LRSN value to the new RESTOREBEFORE parameter that is at a point on the log prior to when the unusable image copy was taken.

MODIFY RECOVERY

Another utility improved by IBM under DB2 Version 9 is MODIFY RECOVERY. The MODIFY RECOVERY utility is used to remove records from the SYSIBM.SYSCOPY table in the DB2 Catalog, along with related log records from the SYSIBM.SYSLGRNX directory table and related information from the DBD.

Before V9, when you ran MODIFY RECOVERY you had to specify the deletion criteria in one of two ways:

1. Given a specific date, delete all recovery information before that data
2. Or given an age in days, delete all recovery information older than the age

But as of V9 we can alternately specify what is to be retained instead of what is to be deleted. This way, we tell the utility what we want to keep and it will remove the rest. So, instead of coding the DELETE parameter (with a DATE or AGE) we would instead code a RETAIN parameter. There are five different options that can be used with the RETAIN parameter:

LAST (*integer***)** – Using this option informs DB2 to choose a date such that the number of full image copies kept is not less than the *integer* specified. So, if we specify LAST(5), then we will delete all entries that exist prior to the date on which the fifth oldest copy was taken. If more than one copy was taken on that date the result could be that more than 5 copies are retained.

LOGLIMIT – Specifying this option will delete all copies where DB2 no longer has log records to recover forward from. DB2 uses the BSDS to establish the oldest archive log timestamp.

GDGLIMIT – You can use this option to enable the number of copies to be retained to be linked to the corresponding GDG base limit.

GDGLIMIT LAST (*integer***)** – It is possible also to combine the GDGLIMIT and LAST options. DB2 will then use the GDG base limit, if the last primary copy is a GDG, if not it uses the integer specified.

GDGLIMIT LOGLIMIT – And finally, we can combine the GDGLIMIT and LOGLIMIT options, too. If the most recent primary full copy is a GDG then the GDG limit is used. If the last copy is not a GDG then the BSDS is used to determine the date prior to which deletions will occur.

For all these options, MODIFY RECOVERY will cause DB2 to determine the most recent date that satisfies the retention requirement. After ascertaining the correct date to use, all entries before that data will be discarded. In effect DB2 chooses the most recent date that would satisfy the retention criteria. So, under some circumstances there may be

more than one image copy on the specific date that is established, and therefore DB2 will keep those additional copies, too.

As an example, say we take 7 image copies on a single day. Then we run MODIFY RECOVERY … RETAIN LAST (3). In this case, all 7 will be retained as they were made on the same day, even though we indicated that we wanted to retain only the last 3.

There is one more significant change made to the MODIFY RECOVERY utility. Prior to V9, information was only deleted when copies were removed from SYSIBM.SYSCOPY. But as of V9, MODIFY RECOVERY will deletes SYSIBM.SYSLGRNX entries even if no SYSIBM.SYSCOPY records are deleted.

This is important because some shops have migrated over to running BACKUP SYSTEM instead of individual image copies. In this case, there will be no individual image copies to remove in SYSIBM.SYSCOPY, but the SYSIBM.SYSLGRNX records will continue to grow. After migrating to V9 you can use MODIFY RECOVERY to trim the size of SYSIBM.SYSLGRNX which can improve performance for processes that need to access log range information.

BACKUP and RESTORE SYSTEM

BACKUP SYSTEM and RESTORE SYSTEM are relatively new utilities, added to DB2 in Version 8. They use disk volume FlashCopy backups and copypool z/OS DFSMShsm V1R5 constructs to copy and restore large volumes of DB2 data. In Version 9 these utilities are enhanced to use new functions available with z/OS V1R8 DFSMShsm.

The biggest Version 9 improvement is the ability to use system level backups (produced by BACKUP SYSTEM) to recover individual table spaces or index spaces. This is helpful because previously you had to recover the entire system, and that is not always what is necessary.

DB2 V9 also delivers the ability for the BACKUP SYSTEM utility to copy the data directly to tape. The new parameters allowing this capability are the DUMP and DUMPONLY options. Keep in mind that directing data to tape will have an impact on the speed of your restore. Restoring from tape will not be as fast as restoring from a FlashCopy made to disk. Of course, having your data on tape can help in terms of storage management, disaster recovery and off-site data storage, and long-term data retention. So be aware of these trade-offs before creating system level backups on tape.

And finally, support for incremental copying has been added to FlashCopy. So now you can take a system level backup and then subsequent incremental system level backups. An incremental FlashCopy will copy only the tracks that have changed on the source volume since the last copy was taken. But unlike a typical incremental image copy, the previous content on the volume(s) will be replaced by the new content. That means there is no merging of incremental copies required; essentially, the merge is part of the incremental FlashCopy.

LOAD and UNLOAD Enhancements

Getting data into and out of DB2 will be easier in Version 9 because both the LOAD and UNLOAD utilities have been improved. The first question people seem to ask with regard to LOAD is "Has performance been improved?" And the answer is yes, due mostly to improvements in handling indexes. IBM reports that performance can improve up to 24 percent if a table has mostly variable keys.

In terms of functionality, the first change we'll discuss is how to deal with DECFLOAT data type in LOAD (and UNLOAD). DECFLOAT is a new data type available in Version 9 with a maximum precision of 34 digits. A decimal floating-point value is an IEEE 754r number with a decimal point.

The DECFLOAT data type is compatible with SMALLINT, INTEGER, BIGINT, DECIMAL, FLOAT, DECFLOAT. Of course, you will have to concern yourself with how the data is handled if it is loaded to or from DECFLOAT. You specify how to handle this manipulation using the DECFLOAT_ROUNDMODE parameter, which can be used in both the LOAD and UNLOAD utilities. Basically, this parameter informs DB2 how to round the data.

Loading and unloading LOBs has been improved, too. For LOAD, an input field value can contain the name of the file that contains a LOB column value. The LOB column value will then be loaded from that file. For UNLOAD, you can store the value of a LOB column in a file and record the name of the file in the unloaded record in the base table.

DB2 Version 9 allows you to skip locked rows in your transactions using the SKIP LOCKED DATA option within your SQL statements. Well, you can skip locked rows when running an UNLOAD, too. If you are running an UNLOAD with SHRLEVEL CHANGE ISOLATION CS, you can also specify SKIP LOCKED DATA. This will cause the UNLOAD utility to skip rows on which incompatible locks are held by other transactions. This option only applies to table spaces with row level or page level locking

RUNSTATS and Histogram Statistics

Another utility upgrade that found its way into DB2 Version 9 /OS is the ability to gather histogram statistics. This article is already quite lengthy, so I won't go into great depth on histogram statistics. Suffice it to say that the ability to collect histogram statistics is a very powerful new capability of the RUNSTATS utility that can be used to gather distribution statistics across all data values. These statistics can be helpful when you need additional distribution data to enable the optimizer to arrive at a better access path for certain predicates and queries.

Improvements to the REORG Utility

The REORG utility has also been improved for Version 9. First of all, let's discuss performance improvements. In Version 9, the REORG utility can unload and reload partitions in parallel. This should result in a nice reduction in elapsed time when you are reorganizing partitioned table spaces. To enable this improvement you will need to code the NOSYSREC keyword or the UNLDDN keyword with a template. Note that NOSYSREC is always used for SHRLEVEL change.

But parallelism will not be enabled if any of the following conditions apply:

- DATAWKnn ddnames are specified in the JCL
- SORTDEVT keyword is not specified
- UTPRINT is allocated to anything other than SYSOUT
- The REBALANCE keyword is used.

Another performance improvement enables REORG SHRLEVEL CHANGE to use subtasks during the LOG phase to speed up the processing of log records. Subtasks will be used when unload and reload have been done by partition and there are log records to apply for the partition.

Perhaps the most anticipated enhancement is the elimination of the BUILD2 phase for Online REORGs. As you probably know, an Online REORG will reorganize using shadow data sets while the data remains available in the primary data sets. When the REORG is done, the shadow data sets are switched to be the primary data sets.

OK, so far, so good. Well, as most DBA should know, reorganization by partition can create a long outage when non-partitioned indexes (NPIs) exist on the table. Why is this so? Prior to Version 9, during Online REORG, NPIs had to be updated with the new RIDs for the data of the partition being reorganized during the BUILD2 phase. And that is what typically caused the sometimes significant outage.

Version 9 eliminates the BUILD2 phase, instead reorganizing the entire NPI and then switching in the SWITCH phase along with all of the other data sets. The only outage is during the SWITCH phase, which will be much, much shorter than the BUILD2 phase outage.

This functional change means that DB2 will need additional temporary storage for the shadow data sets for each NPI. Additionally, the cost of the REORG will increase because building an entire NPI will consume more CPU than the previous method of updating RIDs in the BUILD2 phase. Finally, you will need to modify any REORG jobs against different partitions that ran in parallel because the entire NPI will be built and then switched.

What about running REORG SHRLEVEL REFERENCE by partition? Well, REORG SHRLEVEL REFERENCE by partition will also rebuild any associated NPIs using shadows... so similar concerns apply (more disk storage, additional CPU, no separate jobs in parallel).

REORG has also been improved with regard to its LOB capabilities. Prior to Version 9 you could not access LOB data during a REORG. And a REORG did not reclaim physical space from the LOB data set because LOBs were moved within the existing LOB table space. Version 9 fixes these problems. During a REORG, the original LOB table space is drained of writers. All LOBs are then extracted from the original data set and inserted into a shadow data set. When this operation is complete, all access to the LOB table space is stopped (the readers are drained) while the original data set is switched with the shadow data set. At this point, full access to the new data set is enabled, and an inline copy is taken to ensure recoverability of data.

Miscellaneous Additional Utility Enhancements

Version 9 brings many other utility improvements. One of these changes comes to us with the DSN1LOGP utility. The utility will now detect possible erroneous recovery information. For example, consider a situation where you specify a range of log records to print but the entire range is no longer recorded in the BSDS. This can happen if your archive logs have rolled off. Prior to DB2 V9 DSN1LOGP would have just merrily printed the records for you. But because this can be confusing and could lead you to recovering improperly, V9 will detect this situation, produce a RC 4 and the DSN1224I error message indicating logs could not be found.

Another nice new Version 9 utility-related enhancement is that the DB2 Utilities Panel within DB2I adds an option for you to specify the SDSNLOAD library. This is useful when there are more than one DB2 subsystems on an LPAR.

DSN1COPY has been enhanced, too. As of DB2 9 for z/OS, running DSN1COPY RESET to copy a compressed table space from a non-data sharing DB2 subsystem to another non-data sharing subsystem **will** copy the specified data and reset the PGLOGRBA of each page and the dictionary version. In V7 and V8 the dictionary version was not reset (instead, you had to REORG the copied table space with KEEPDICTIONARY=NO in the target subsystem). This was not an issue for data sharing because the LRSN is used instead of an RBA.

CATMAINT is also enhanced in Version 9 to allow modification of VCAT details, owner, creator, and schema names. It also provides an option to change ownership of objects to a role. Roles are new in Version 9, too.

Summary

IBM has cooked up a nice stew of new utility functionality for us in DB2 Version 9. Be sure to review and understand all of these new capabilities as you migrate to the new version.