

DB2 Database Maintenance and Recovery:

Getting Past the Common Misconceptions

By Brenda Honeycutt and Craig S. Mullins

Every DBA strives to maintain a healthy DB2 environment every day. However, there are numerous lingering misconceptions about database health to which many organizations fall prey. These misconceptions occur because DBAs believe in, or apply, concepts that are objectively false.

Some misconceptions exist simply because they've been passed down from older generations. In terms of DB2, this means that best practices for older DB2 versions may no longer be best. This article examines several common misconceptions about DB2 maintenance and recovery.

"Lies, Damned Lies, and Statistics"

This is part of a phrase usually attributed to Benjamin Disraeli and popularized in the U.S. by Mark Twain. Although the origin of this statement may be questioned, its meaning is clear. It refers to the persuasive power of numbers and neatly describes how even accurate statistics can be used to bolster inaccurate arguments.

When it comes to DB2, statistics are the lifeblood of database health. But it's important to distinguish between the two types of statistics available to us: catalog statistics and real-time statistics. Although originally intended as food for the DB2 Optimizer, historically we've also needed analytical RUNSTATS that updated catalog statistics to drive our maintenance processes. With the advent of real-time statistics, there's no need to waste time and resources by running analytical RUNSTATS. It's time to use RUNSTATS strictly for the purpose IBM intended: driving access paths for optimal performance.

Let's dispel the delusion that using real-time statistics increases CPU costs. DB2 collects the statistics in real-time anyway, even if you aren't using them. Experience and customer polls indicate that externalizing RTS has practically no impact on CPU. Because these statistics are collected in real-time, they're timely and accurate.

So you have your statistics straight. They accurately reflect your database activity and you can rely on them as a basis for backups. The real-time statistics that pertain to backups include:

- **COPYLASTTIME:** The timestamp of the last full or incremental image copy on the table space or partition
- **COPYUPDATEDPAGES:** The number of distinct pages that have been updated since the last COPY

- **COPYCHANGES:** The number of insert, delete, and update operations since the last COPY
- **COPYUPDATELRN:** The LRSN (Log Record Sequence Number) or RBA (Relative Byte Address) of the first update after the last COPY
- **COPYUPDATETIME:** The timestamp of the first update after the last COPY.

Database Backup Fallacies

Now let's consider some common fallacies regarding database backups outside the realm of determining when to do them:

Migrating too early to tape. Some organizations immediately migrate current backups to tape to free up disk space. This will free up some DASD, but the time needed to recall that tape must be added to the recovery time. So which is more important: disk space or recovery time?

You should consider keeping the most current backups on DASD and migrate them to tape only when a new backup is taken. Of course, your tape vs. disk strategy for backups will heavily depend on the nature of the data being backed up and Recovery Time Objectives (RTOs) for that data.

Some shops use dual backups. Rather than trying to save money on disk space, consider limiting dual backups to objects that require a critical copy. A critical copy is one taken after a REORG or a LOAD LOG NO, as well as a REORG with a rebuild of the compression dictionary for compressed table spaces. In other cases, DB2 always has the opportunity to fall back to a prior copy in the unlikely event of an unusable copy.

Virtual tape storage is as good as DASD. Not. There's a hard-and-fast rule concerning Virtual Tape Systems (VTS) that everyone should follow: Although choosing a less expensive storage media, such as VTS, for backups is OK, don't overestimate the speed of VTS. When deciding on the right backup device, always keep mount and recall times in mind. For example, mount and recall times for small objects are often bad in relation to recovery time.

Oops, the archive logs are on tape! If your archive logs are on tape or VTS, a parallel recovery isn't possible. Keeping archive logs on DASD isn't an option for some shops. When backing up archive logs to tape, don't follow the rule of thumb pertaining to the 28,672 block size normally optimal for tape. Instead,

use a block size of 24,576. To enable parallel processing for recovery, the archive logs on tape must be copied to DASD, and 24,576 is the preferred size; it's been the ZPARM default since DB2 V8.

Think about the trade-off in tape savings vs. recovery speed and your decision should be obvious. Before backing up any logs to tape, it's a good idea to always have at least 24 hours covered by the active log and at least 48 hours covered by archive logs on DASD.

You can use mirroring instead of dual logging. Some organizations mistakenly believe that single active logging is fine if mirroring is used. But what happens if a technical error occurs with the channel, the device, or the firmware of the device? The data will be mirrored as inconsistently as the source. You should always do dual logging, *without exception*. The DB2 log is the most important resource to ensure data consistency.

Fast, faster, flash it. IBM introduced BACKUP SYSTEM and RESTORE SYSTEM in V8 to take full advantage of the immense speed gains in FlashCopy I and II technology. With FlashCopy, the I/O subsystem does the copy, freeing the CPU to do more important work. FlashCopies enable ultra-fast image copies and now support incremental copies at the volume level and over multi-volumes when using "consistency groups."

Some people have hailed FlashCopy as "the end of the image copy," but that isn't the case for all installations. It's the best for select, large, enterprise-critical objects, but you should choose those objects wisely. FlashCopy can be expensive in terms of the hardware cost and system topology. You must have twice the number of disks to flash a given "pool," which can make keeping generations of backups extremely expensive. Even if you then offload the flash pool to tape, you fill up tapes with gigabytes of unchanged data. The introduction of incremental FlashCopies has alleviated some, but not all, of this space management nightmare. Incremental flashes are at the volume level. A data set that's multi-volume must be handled with great care—unless persistent, change recording, background nocopy, and inhibit target write options are all in effect.

FlashCopies aren't registered anywhere, unless from the BACKUP SYSTEM DB2 command. You must

remember that you did a flash, and remember what was flashed. Flashes don't reset the COPY PENDING status. Don't, however, let these facts discourage you from using FlashCopy outside of DB2. Some third-party tools keep track of FlashCopies and reset the copy pending status. Maybe yours is one of these.

Copying indexes. Some folks blindly continue on, administering their DB2 systems with little or no changes even as new versions roll in with new functionality. We've used COPY to make image copy backups of DB2 indexes for some time now, but many shops completely ignore this capability. If the index was created (or altered) with the COPY YES parameter, then it can be recovered using either the REBUILD or RECOVER utilities. An index defined as COPY NO can only be recovered using the REBUILD utility.

It can make sense to backup indexes for recovery because the REBUILD utility can take a long time to complete for large amounts of data or when a large number of indexes exist. For example, in one simple test, recovering a 50-million row index substantially outperformed rebuilding that same index. Of course, your mileage may vary.

Another reason for backing up indexes is in the case of a recovery of both the data object and the index. COPY YES indexes can be restored at the same time as the data is being restored, reducing the overall outage time.

Remember, however, that all index copies must be full copies, as incremental image copies aren't permitted for indexes.

Recovery: Ready, Set, Go

If you prepare well, there should be no recovery problem since all backups are available and current. You're carefully aligning your backup frequency to update rates using automation, thresholds, and monitoring. You're likely convinced your backup strategy is fine-tuned.

What's wrong with this picture? You probably haven't the slightest inkling of whether the recovery duration supports the business needs. Threshold-based backups simply neglect RTOs. A look at two scenarios found in most shops best illustrates this.

There's a misguided notion that recovering small objects, especially those with minimal updates, will always be fast. But this doesn't consider situa-

Tips

- Use real-time statistics to eliminate analytical RUNSTATS.
- Don't migrate backups too early to tape.
- Limit dual copies to critical copies.
- Virtual tape storage isn't as good as DASD.
- Use a block size of 24,576 if you must place archive logs on tape.
- Always use dual logging.
- Choose objects for FlashCopy wisely.
- Use the COPY YES when creating or altering indexes.
- Consider RTOs when creating backups.
- Know your ZPARMS well.
- Avoid multi-volume data sets.
- Check for a healthy CF.
- Don't rely on logging for auditing purposes.

—BH & CM

tions where single updates to a page cover several archive logs. Recovering a large object with high update rates, all of which are active, can be much faster.

On the other hand, what about those large objects with high update rates that contain your most critical enterprise data? Although a well-adjusted procedure to back them up every night is helpful, large volumes of transactions during the day may lead to update rates being exceeded by midday. Being forced to recover using excessive log applies leads to a significantly increased recovery time. An intelligent use of real-time statistics combined with, for example, IBM's online COPY, will ensure just-in-time backups that support business availability requirements.

In terms of recovery-time objectives, consider several other factors such as ZPARMS, use of multi-volume data sets, and the Coupling Facility (CF).

A, B, Z's of ZPARMS. Don't depend on ZPARM defaults—or any defaults, for that matter. Defaults might seem to make sense for “one size fits all” shops, but even if your shop is an average fit, the recommended values change with new versions of DB2. Migrating to a new version doesn't apply the new defaults.

Fast Log Apply (LOGAPSTG) represents the log apply storage and should be set to the maximum of 100MB. Although most tuning experts agree with the old maxim “never say never,” there are exceptions to every rule; so never define a value of less than 100MB for LOGAPSTG. This is the default value, but it was changed only since V8. If DB2 gets less, it just takes less. Limiting this value critically restricts the available log apply storage and may slow down DB2 restart or a recovery. Assigning anything less than 100MB just doesn't make sense.

The check frequency parameter (CHKFREQ) determines the system checkpoint frequency in minutes or in number of log records. The default is 500,000 log records. Consider changing this to a time interval to maximize performance. Using log intervals can miss important checkpoints if your logging rates significantly vary. A reasonable starting point is to set CHKFREQ at two to five minutes.

If data sharing is in use, Retained Lock Wait (RETLWAIT) represents a multiplier on how long a member of a data-sharing group waits for locks on a resource held by another member that has failed. The multiplier is applied to the connection's normal time-out value and should be set to 2. The default is 0, which means the lock request is immediately rejected and the application receives a resource unavailable SQLCODE.

The OUTBUFF parameter is the size of the output buffer that's used for writing active log data sets. Many shops are still using the V7 default of 400, which was changed to 4,000 for V8. To decrease the number of forced I/O operations that occur because there are no more buffers, you should select as large a size as your system can tolerate. Some popular health checks recommend a minimum value of at least 40,000K to increase the speed that DB2 can roll-back to. Reading the output buffer is

always much faster than reading the active log.

These are only a few examples of relevant ZPARMS—the ones that impact your recovery times. The bottom line is to know your ZPARMS well and, because no installation is static, always do periodic checks on the values you choose. The best resource for DSNZPARM information is the *IBM DB2 Installation Guide* (GC18-9846-03).

The CF performs great. Don't be fooled! When a member or a CF comes crashing down, all the data sets are put into LPL or GRECP. To get the objects out of LPL or GRECP, you'll have to build and submit lots of jobs. Each job should contain up to 99 -STA commands and up to 10 jobs can run in parallel. Why all these odd numbers? It boils down to LOGAPSTG being used as much and as efficiently as possible. Once you've built and submitted all these jobs, the CF and the SCA, which holds the LPL, starts thrashing for its very life. The following guidelines are indicative of a CF that will stand the test of a speedy recovery:

- “CF transfer time” is less than 2,000 microseconds.
- “Number of messages rejected due to lack of message buffer” is zero.
- “Sub channel busy” percentage is less than 10 percent.
- “False lock contention” percentage is less than 3 percent.
- “All path busy termination count” is zero.
- Group Buffer Pools (GBP) “cross invalidations due to directory reclaims” is zero.
- GBP “asynchronous failed writes due to lack of storage” is zero.

The GBP are easy to check with a DISPLAY command. For GBP problems, use of the “auto alter” in z/OS is a good starting point. The others, however, require Resource Measurement Facility (RMF) reports for false lock contention and deep knowledge of system internals such as CF Assembler macro Application Program Interfaces (APIs), of which there are many. The following macros will get you started:

- IXCQUERY requests information about resources the CF manages.
- IXLIMG requests measurement data related to the use of a CF.
- IXLZSTR retrieves control information and structure data from a dump.

When it comes to DB2 maintenance and recovery, what you don't know—and especially what you think you know—can hurt you.

If the CF metrics are obscure or exceeded, you may well be heading for real trouble when you attempt a recovery.

Multi-volume data sets rule. Often, and perhaps without a DBA's awareness, DB2 table spaces are spreading out over an unforeseeable number of volumes. If this gets out of control, kiss your RTOs goodbye while you end up fiddling with multi-volume data sets before the real recovery activities can even start.

Quite regularly, application problems become database administration problems, which in turn become DASD space management problems. It's not uncommon for a DBA to knock on the door of the Space Management Department to ask for relief. Make sure the "help" you get doesn't worsen your problems! Over the last decades and years, DASD space management has advanced considerably. SMS can offer so much relief that one DATA CLASS and parameter was even named after this major effect: space constraint relief.

Yet space problems are space problems. If there's a permanent shortage of DASD space, even the most sophisticated settings of SMS DATA CLASS parameters aren't going to help. Pay close attention when somebody starts mumbling terms such as volume count, space constraint relief, dynamic volume count, or extent constraint removal. The settings for those parameters determine whether a data set or cluster will be multi-volume.

If you set DATACLAS to enable multi-volume allocations (SPACE

CONSTRAINT RELIEF = Y and DYNAMIC VOLUME COUNT to more than one), then reorganize everything, you could end up with thousands of small, multi-volume data sets that just soak up below-the-line storage and make I/O, especially OPEN and CLOSE, slow. This is especially true of highly fragmented disks.

Only hand-picked, large VSAM clusters should be multi-volume. They must be clearly documented, and they must be part of your recovery planning and recovery procedures.

VSAM extent constraint removal was introduced with z/OS 1.7. It removes the 255-extent limit (255 extents per component, or per stripe, respectively), whereas the limit of 123 extents per volume remains. The maximum number of extents is now 7,257—59 volumes with 123 extents each. You don't want that, do you?

On Logging and Auditing

Probably one of the biggest misconceptions is that logging supports auditing requirements. Transaction logs are an integral component for ensuring database integrity. Logs function to record all changes as they're made to DB2 data. DB2 will log every modification made to every piece of DB2 data. Log records are written for every modification that's successfully executed and committed (with a few exceptions, such as LOAD LOG NO). Log records are written to the active logs. There are usually two active log data sets to safeguard against physical device errors. The active logs must reside on disk; they can't reside on tape. DB2 manages the active log data sets using the BSDS.

As the active logs are filled, DB2 invokes a process called log offloading to move the log information offline to archive log data sets. This process prohibits the active logs from filling up during DB2 processing, which would stifle the DB2 environment.

The logged data can be used for rolling back transactions and recovering table spaces and indexes. Indeed, this is its primary and most important purpose—to ensure the integrity of the data in your DB2 databases. However, some organizations use the logs for database auditing.

Database auditing is the process of monitoring access to and modification of selected database objects and resources within operational databases and retaining a detailed record of the access where that record can be used to proactively

trigger actions and be retrieved and analyzed as needed. But it's a bad idea to rely on the transaction logs for your auditing.

The biggest problem when relying on logs for auditing is that reads aren't recorded. Sometimes regulations and policies require auditing reads, but if you need to know who accessed a particular table and when, the log can't answer that question.

Another problem with using logs for auditing is a lack of separation of duties. Understanding what's on the logs and which logs are needed requires DBAs. But we also must audit DBA activity. This is a scenario auditors don't like.

Perhaps the biggest problem with log-based auditing, however, is one of volume. Database auditing should be performed only on those database objects containing sensitive information. You don't want to audit everything. But the DB2 logs are non-discriminating; that is, every change to every database object is logged. If you rely on the logs for auditing, the resulting mountain of logs can quickly become an administrative burden.

To do database auditing justice requires a software solution that captures database requests from the server without impacting database logs or recovery objectives.

Summary

Recognizing misconceptions is the first step to defeating them. After that, you can find better ways to perform your processes. Today's business availability requirements are higher than ever; unless you've already been forced into a corner, you may have been lucky enough not to even know that some practices aren't the best ones. Lots of resources and tools are at your disposal. By separating fact from fiction, you can help maintain a healthy DB2 environment. **Z**

About the Authors

BRENDA HONEYCUTT is a DB2 software consultant at Software Engineering GmbH, where she also manages the technical publications department. Software Engineering is a DB2 solutions provider specializing in tools and consulting for more than 20 years. Email: b.honeycutt@seg.de

CRAIG S. MULLINS is a data management strategist for NEON Enterprise Software, Inc. He is a frequent blogger, speaker, and author, and is involved with several IT industry organizations, including co-chairing the 100 Year Archive Committee in the Storage Networking Industry Association. Website: www.craigsmullins.com