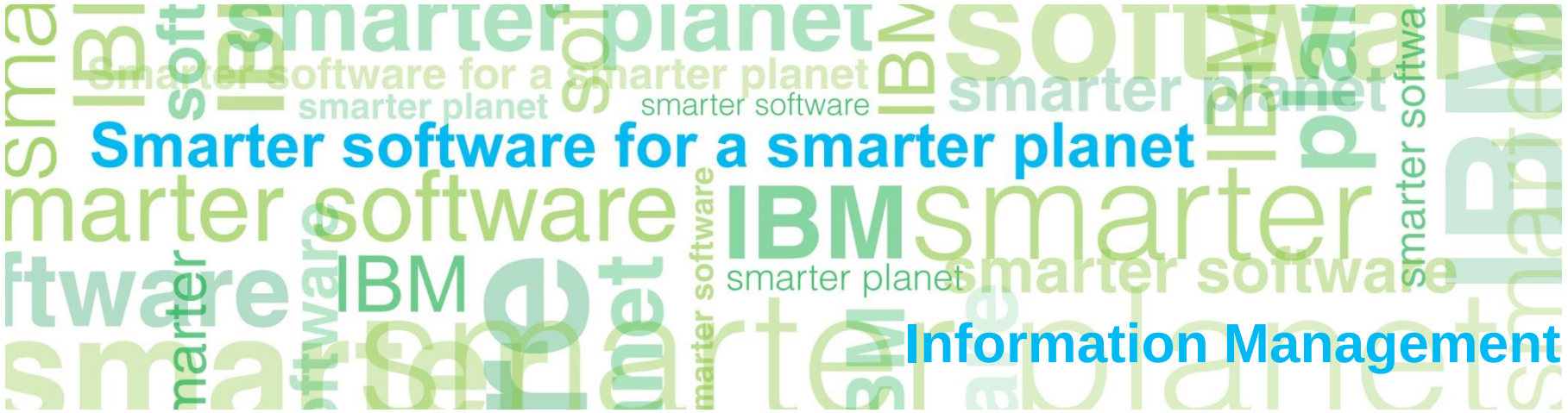
Robert Catterall, IBM
rfcatter@us.ibm.com

DB2 for z/OS Backup and Recovery: Basics, Best Practices, and What's New

Baltimore / Washington DB2 Users Group

June 11, 2015



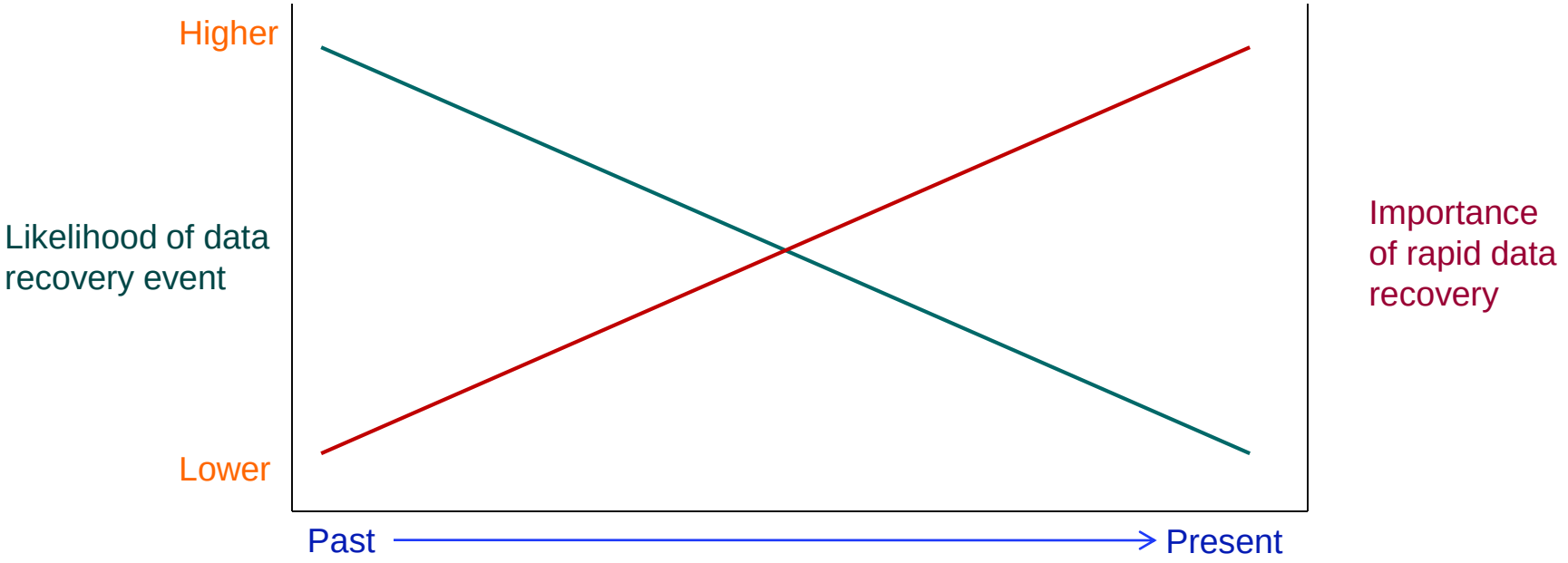
Agenda

- Data backup and recovery – more important now than ever
- Backup and recovery basics
- Pitfalls – things that some folks get wrong
- Some best practices
- Backup and recovery-related enhancements delivered with DB2 10 and 11 for z/OS

Data backup and recovery – more important now than ever

Recovery: need less likely, speed more important

- With the growing reliability of hardware and software, data recovery events are more uncommon…



- …but growing intolerance of application downtime, and growing value of data, put a premium on fast recovery

Data unavailability less likely, more costly

- Some reasons why data recovery events are more uncommon:
  - RAID-based disk subsystems (redundancy of data storage)
  - More solid-state storage devices (literally, fewer moving parts)
  - Greater use of automated operations, and more autonomics built into key subsystems such as DB2 for z/OS (reduces chances of human error)
  - Pervasive use of online DB2 REORG (if it fails, only “shadow” objects affected – “originals” not in RECOVER-pending state
- Some reasons why data unavailability events are more costly:
  - If your Web site is down, you could lose customers – permanently
  - With tight supply chain management, an outage could mean out-of-stock
  - With “wired” factories, outage could mean idled assembly lines
  - With securities trades, if outage delays trade execution, your company is on the hook for any resulting losses

And yet, backup/recovery is under-appreciated

- Not as “sexy” as DB2 for z/OS and application performance tuning

- But consider this: an application isn’t performing very well when it’s down

So, pay attention to DB2 backup and recovery

Backup and recovery basics

Nothing is more important than being prepared

- If a data loss event COULD occur, you must assume that it WILL occur – be ready for it
  - Practice your data recovery procedures
- On top of that, don't get too set in your ways
  - Review your DB2 data backup and recovery processes and procedures on a regular basis – annually, at least
    - Are you taking advantage of backup- and recovery-related enhancements provided by z/OS, DB2, and storage systems?
    - Are you aware of data in your DB2 environment that has become more – or less – critical over time? Do you know what needs to be recovered first?
    - Is the documentation of your DB2 data backup and recovery processes and procedures up to date – would you lose critical knowledge if someone on your team were to leave your organization?

Some basics

- For objects that are not read-only, do at least an incremental image copy on a nightly basis
  - Full image copies can be generated weekly
- Merge incremental image copies into full image copies
  - So DB2 won't have to do that in the event that a recovery is required
- Keep at least 24 hours of data change information on the active log data sets of a production DB2 subsystem
  - To reduce the likelihood that archive log data sets will be needed for a data recovery operation

More basics

- Partition larger tables
  - So that you can recover individual partitions, versus having to recover the whole thing
  - One of the great things about partition-by-growth universal table spaces: they deliver backup/recovery benefits of partitioning for large tables that do not have a good range-partitioning key
- Run the MODIFY RECOVERY utility regularly…
  - To clear old and unneeded backup records from the SYSCOPY table in the DB2 catalog
- …but see that you retain the records for the two most recent full image copies of every object that you back up
  - The RETAIN option of MODIFY RECOVERY can help in this regard

Pitfalls – things that some folks get wrong

Don’t do unnecessary stuff (1)

- Avoid executing unnecessary image copy jobs
  - At some sites, all table spaces are image copied on a regular basis, including those that have not had any update activity since they were last backed up
    - Wastes CPU cycles, disk space (if you copy to disk – more on that later)
  - Table spaces unnecessarily backed up are often of the historical or archival variety
  - Best way to avoid this pitfall: build intelligence into backup procedures
    - Could be accomplished through user programming (often in REXX), along with information from SYSTABLESPACESTATS catalog table (such as COPYUPDATEDPAGES, the number of pages updated since last COPY)
    - Could be accomplished with user programming + IBM-supplied DSNACCOX stored procedure, which uses real-time statistics information to make (among other things) image copy recommendations for database objects
    - Could be accomplished with vendor-supplied DB2 utility automation tool

Don’t do unnecessary stuff (2)

- Don’t run DB2 QUIESCE utility if you don’t need to
  - In the past, QUIESCE was routinely run to establish a log point to which a table space (or table spaces) could be recovered, with consistency
    - Consistency in this context meaning that data in the table space (or spaces) is not affected by any partially done units of work
  - Starting with DB2 In new-function mode, the RECOVER utility can recover an object (or a set of objects) to ANY log point and ensure consistency, even if that log point was not established via QUIESCE
    - How that’s done: RECOVER detects any incomplete URs associated with the target table space(s) at the designated recovery log point, and backs out any changes made by those incomplete URs
  - This change to RECOVER should significantly reduce the need to run the QUIESCE utility
    - And that’s good, because as DB2 for z/OS workloads have gotten bigger, it’s become harder to run QUIESCE without disrupting applications

A little more on QUIESCE

- A unit of work might change data in more than one table space – keep that in mind if you're counting on RECOVER's ability to back out in-flight URs at any log point
  - If recovering table space XYZ to a prior point in time, include in the RECOVER job any table spaces also updated by programs that update XYZ
- Still might run some QUIESCE jobs, especially if that can be done in a non-disruptive fashion
  - Can be useful for establishing a recovery point generated just before the start of a data-changing batch job
  - Can speed up execution of RECOVER (no need to back out incomplete URs if recovery log point was generated via QUIESCE)

Don’t over-rely on point-in-time recovery

- These days, many (maybe most) organizations do not have a true batch window (i.e., a time when only batch jobs are running) – transactions and batch jobs access DB2 tables
  - Point-in-time (PIT) recovery has speed and simplicity advantages when it comes to recovering from data corruption caused by a batch job logic error or a bad input file (recover to point just before job started)
  - Transaction access to tables that are accessed by a batch job can make PIT recovery unfeasible
    - If transactions are updating the same tables that are updated by a batch job, PIT recovery will undo the “good” changes made by transactions along with the “bad” changes made by the batch job
    - In that case, undoing batch job-caused data corruption may require a DB2 log analysis tool (available from IBM and other vendors) that can selectively back out changes made by a particular batch job

Don’t write DB2 archive logs to tape…

- …at least, not initially
- Problem with tape is that it becomes a serializer (in other words, a bottleneck) when recover jobs are run in parallel and multiple jobs want to read from same archive log data set
- Better idea: write archive log data sets to disk, then migrate them to tape via HSM (but let them stay on disk for 2 or 3 days before you migrate them to tape)

Some best practices

Prevent accidental dropping of objects

- Use RESTRICT ON DROP (option of CREATE and ALTER TABLE) for tables in production DB2 subsystems
  - With that option in effect, dropping table can only be accomplished in one of these two ways:
    - Execute the REPAIR utility with the DROP DATABASE option (not recommended)
    - Issue an ALTER TABLE statement with DROP RESTRICT ON DROP, then drop the table the normal way

Image copy indexes on larger tables

- Do-able since DB2 V6
  - But not used as much as it should be
- Advantage: for a large table, there's a good chance that RECOVER of an index will be faster than REBUILD of the same index
- Advantage: if you lose a table space AND its indexes, you can RECOVER indexes WHILE table space is being recovered *if indexes were image copied*

  If you rely on REBUILD INDEX, you will have to complete recovery of the table space before you can start the index-rebuild process

Consider range-partitioning large tables by date

- At one time, this was not recommended, because partitioning by a continuously-ascending key was not recommended
  - Reason: you were limited to 254 partitions, and there was concern that you'd "hit the wall" (i.e., the last partition) sooner than desired
- That changed with table-controlled partitioning (DB2 V8)
  - Up to 4096 partitions
  - You could have a week of data in each partition, and after 20 years you'd have used just over 25% of the partition limit
    - ALTER TABLE DROP PARTITION is a known requirement – I think that DB2 development will deliver that before you run out of date-based partitions
- If only data in "current" partition is updated, no need to repeatedly image copy older partitions

Some backup and recovery-related enhancements delivered with DB2 10 and 11 for z/OS

Dynamically add active log data set via command

- Before DB2 10: if there is trouble with log archiving, and the active log data sets aren't being offloaded as needed and they fill up, you've got a real problem
  - The DB2 subsystem essentially grinds to a halt
  - To give yourself some breathing room to address the archiving difficulty, you need to add one or more additional active log data sets
  - To add active log data sets, you had to run the DSNJU003 utility
  - To run DSNJU003, DB2 must be down
  - To shut down properly, DB2 needs to write some information to the active log
  - **BUT THE ACTIVE LOG IS FULL!!!**

DB2 10 to the rescue…

- New option for the -SET LOG command: NEWLOG
  - Allows you to add a newly defined data set to the active log inventory
  - Assuming that DB2 can open the data set, it will be immediately available for use, with no need to recycle the DB2 subsystem\
    - So, you first define the new active log data set with IDCAMS
    - Recommended that you also format the new data set via the DSNJLOGF utility, so that DB2 won't have to do this after opening the data set
    - Issue the -SET LOG NEWLOG command twice to add a new active log data set pair (LOGCOPY1 and LOGCOPY2), if you're doing dual logging

Another DB2 10 feature: backward log recovery

- Consider this scenario:
  - You image copy a table space
  - 23 hours later a batch job kicks off and inserts some bad records into the table space
  - You want to recover the table space to a point in time just before the bad-insert batch job started
- Given that scenario, what could you do in a pre-DB2 10 system?
  - You could run a standard RECOVER job
    - DB2 would restore the most recent image copy of the table space (produced 23 hours ago) and would apply logged data changes from that point forward until reaching the desired recovery point

DB2 10: BACKOUT YES

- New option for RECOVER TABLESPACE UTILITY
  - When specified, RECOVER will start with the target object in its current state (there's no restore of an image copy) and will back logged data changes out until the desired recovery point is reached
    - And if there are any incomplete URs associated with the table space at that recovery point, RECOVER will back those out to ensure that the data in the table space is consistent
  - Result can be significantly reduced RECOVER elapsed time

DB2 11: PIT recovery and pending DDL

- With DB2 10, if pending DDL changes for a table space are materialized via an online REORG, you cannot recover the table space to a point in time prior to that materializing REORG
- With DB2 11 in new-function mode, that restriction is partially removed
  - You can recover a partition-by-range universal table space (or a LOB table space or an XML table space) to a point in time prior to an online REORG that materialized pending DDL changes for the table space
    - The pending DDL-materializing online REORG must have been run in a DB2 11 NFM environment
    - Following the recovery to the PIT prior to the materializing REORG, the table space will be in a “hard” REORG-pending state (REORP), and a subsequent REORG will be required to make the data available to applications

And that's all – thanks for your time

Robert Catterall
rfcatter@us.ibm.com