Acknowledgements and Disclaimers:

**Availability.** References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates.

The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS-IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

© Copyright IBM Corporation 2011. All rights reserved.

- **U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.**

IBM, the IBM logo, ibm.com and DB2 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.
Agenda

- Overview
- REORG Processing
- NPSIs
- Drain processing
- Controlling Availability
- Other V10/V11 REORG enhancements
- PBG
- LOBs
- What and When to REORG
- Summary of Recommendations & Best Practices
- Summary
Overview
Why REORG?

- **The REORG utility is critical for core function enablement in DB2 for z/OS**
  - Non-disruptive materialization of online schema changes
    - Added columns
    - Altered columns
    - BRF-RRF row format conversion
    - Inlining of existing LOB data
    - Conversion to/from hash pageset format
    - Partition limit-key changes
    - Conversion to UTS
    - Page size alteration
    - Extended page format conversion
    - Redistribute data across partitions, even with LOB columns
    - Discard of data from tables
    - Resize pagesets
    - Resize hash space, including auto-resize
    - Compress data
Why REORG?

• **And of course... REORG for application performance**
  • Re-chunk LOB data
  • Re-establish data clustering
  • Re-establish free space
  • Consolidate pointer-overflow records
  • Remove deleted & pseudo-deleted records
  • Reorder index leaf pages & remove pseudo-deleted index entries

• **All this, whilst maintaining application availability**
REORG Processing
REORG SHRLEVEL CHANGE phases & availability

- SHRLEVEL CHANGE & DRAIN ALL
- Not to scale
Phase processing

- **UTILINIT**
  - Build data structures
  - Serialize objects
  - Create shadow pagesets

- **UNLOAD**
  - Unload pagesets in parallel
  - Pass to sort & start data sort

- **RELOAD**
  - Continue sort process
  - Load sorted data into shadow pagesets
  - Build inline image copy
  - Gather statistics if requested

- **SORTBLD**
  - Concurrent with load process
  - Extract index keys, sort if necessary & build shadow index pagesets
Phase processing

- **LOG**
  - Scan log in multiple iterations for missing data updates & apply to shadow pagesets
  - Continue inline image copy build
  - Drain & complete log phase

- **SWITCH**
  - Finish inline image copy
  - Switch shadow & main pagesets
  - Update catalog & directory information

- **UTILTERM**
  - Release drain
  - Update catalog statistics if requested
  - Delete shadow datasets
  - Deserialise & clean up
Required System Resources

- **DASD requirement**
  - Shadow data sets
  - Sortwork data sets
  - Work data sets (e.g. SYSUT1, SYSREC)
  - Image copies

- **Sorting**
  - Data sort into clustering order
  - Index key sort into key order
  - Inline statistics sorts for non-distribution statistics (e.g. COLGROUP, DPSI)
  - Supports both DFSORT and DB2 Sort

- **Other**
  - Mapping index for RID translation (SHRLEVEL CHANGE only)
  - Active log (for data and/or index log apply)
NPSI

Smarter software for a smarter planet

Information Management
NPSI processing with partition-level REORG TABLESPACE

- **Prior to V9**
  - No shadowing of NPSIs
  - BUILD2 phase at end of REORG leaves logical NPSI partitions unavailable

- **V9 & beyond**
  - BUILD2 phase removed
  - NPSIs shadowed

- **Implications of V9 removal of BUILD2 phase**
  - Better availability since NPSI processed whilst pagesets are UTRW
  - Shadow of NPSI means cannot REORG parts in separate REORG jobs concurrently
  - Entire NPSI must be drained for switch to shadow
  - Processing of entire NPSI can be costly
NPSI processing cost

- 2 parts to NPSI cost when reorging subset of partitions
  - Keys for partitions not included in REORG are unloaded in key order to build shadow
    - Can take hours to scan thru a large, disorganized NPSI
    - Improved performance in DB2 V10 with index leaf page list prefetch
  - Once sorted, keys extracted from data being reorged are inserted into shadow NPSI
    - Can result in index leaf page splits
    - Potentially exhaust the previously reserved free space in the index
  - Cost of 1 is higher if NPSI is disorganized since keys extracted in key order
  - If small % of data is reorged then cost of 1 is higher and cost of 2 is lower
  - If high % of data is reorged then cost of 1 is lower and cost of 2 is higher
- Consideration:
  - REORG of entire table space may be cheaper than REORG of subset of parts
  - Consider converting NPSIs to DPSIs – but be aware of potential application performance impact of DPSIs
Improve performance of part-level REORG with NPSIs

- New option to defer shadow index build until all keys passed through sort
  - Trade off between the processing cost of sorting all the keys versus driving singleton key inserts into the shadow NPSI
  - The more data records being reorganized, the higher the cost of singleton key inserts
- New keyword (SORTNPSI) & zParm (REORG_PART_SORT_NPSI) to govern
  - AUTO/YES/NO options
  - Keyword overrides zParm value when specified
- Retrofit to DB2 9 & 10 in PM55051
- Result:
  - Customer test of REORG of 40% of partitions showed 55% ET reduction & 22% CPU increase
  - DB2 Sort gives additional ET reduction & cuts CPU to less than original starting point
Improved REORG LISTDEF processing

- PARALLEL YES/NO option introduced in APAR PM25525 in V9
  - NO – Prevent REORG from processing multiple partitions in single REORG when input is partlevel LISTDEF
  - Zparm REORG_LIST_PROCESSING
- Need compromise option for customers who want to take advantage of REORG parallelism but cannot afford to shadow many partitions at a time
- New option LISTPARTS n to limit # of partitions to be processed in a single REORG if input is a part-level LISTDEF
- In DB2 11, PARALLEL YES/NO is superseded by LISTPARTS n
  - PARALLEL YES/NO IS deprecated but still supported in 11
- Recommendation:
  - For optimal performance, REORG all target partitions as a single group whenever possible
  - When constrained by system resources, use LISTPARTS to break up the reorganizations into manageable chunks
DRAIN Processing

Smarter software for a smarter planet

Information Management
Drain processing

- Drain of all claimers required to perform switch
- Pagesets drained serially
  - So REORG of multiple parts can require extended period for drain completion
- 2 options:
  - DRAIN WRITERS or DRAIN ALL
- Why is DRAIN ALL recommended?
  - With DRAIN WRITERS, failure of subsequent read claims causes unnecessary impact to write claimers that have already been drained
  - Undetected deadlock with applications that claims read and then claims write
SWITCH phase impact relief – reduced application impact

- Easier drain acquisition
- Prevent new claims on all target partitions whilst waiting for drains
  - Faster drain acquisition for part-level REORG
- New DRAIN_ALLPARTS option to momentarily drain all data parts
  - Eliminates claim-drain “deadlocks” for part-level REORG with NPSIs
- Restructure SWITCH phase processing for outage reduction
  - SWITCH phase ET reduction of 91% measured when reorging 20 parts
Timing of SWITCH phase with MAXRO DEFER

- Govern timing of drain and switch for long-running REORGs without the need to schedule separate –ALTER UTILITY command
- New SWITCHTIME parameter to determine earliest point at which drain processing will be attempted
Controlling Availability

Smarter software for a smarter planet

Information Management
What parameters govern drain processing?

• **When do I try to drain?**
  • MAXRO = estimated duration of the last log iteration
  • Uses previous log iteration to estimate time taken to process to end of log
  • Alterable with –ALTER UTILITY command
  • MAXRO DEFER will keep REORG in LOG phase until further notice

• **How long do I wait for the drain to succeed before giving up?**
  • DRAIN_WAIT

• **If drain fails, should I try again, and if so, when?**
  • RETRY & RETRY_DELAY
Controlling availability

- Outage duration is from time drain is requested to the time the drain is released
- Application claims will wait and may time out based on IRLMRWT
- Duration encompasses
  - Time required to drain
  - Last log iteration in LOG phase
  - SWITCH phase
  - Start of UTILTERM phase prior to dedrain

- Total outage duration is not just SWITCH phase or MAXRO time
  - It is from the time to drain in the LOG phase to when REORG dedrains after SWITCH
  - Hence recommendation is: MAXRO + DRAIN_WAIT + SWITCH duration < IRLMRWT
How to control outage duration

- **Reduce time required to drain**
  - REORG at a quiet time
  - Have well-behaved applications that commit frequently
  - Have fewer pagesets in single REORG since processed serially
  - FORCE option in DB2 10

- **Reduce duration of last log iteration**
  - Reduce MAXRO
    - But if too short and high logging rate then REORG may never try for drain
  - Use inline Flashcopy instead of inline sequential copy
    - Eliminate cost of final incremental processing during last log iteration
  - Reduce number of pagesets processed in single REORG
How to control outage duration

• **Reduce SWITCH phase duration**
  • Use inline Flashcopy instead of sequential copy
  • Reduce number of datasets processed by single REORG
    • Switch phase duration is dependent on number of datasets being processed

• **Reduce UTILTERM processing prior to dedrain**
  • Move to DB2 10
    • DB2 10 updates catalog statistics columns after dedraining
    • Or gather less inline statistics
    • Or reduce number of datasets processed by REORG
Controlling REORG completion

- Set high MAXRO
  - “Guarantees” drain will be attempted

- Set high DRAIN_WAIT
  - “Guarantees” drain success

- Once drain is successful, no other parameters can prevent REORG completion
Other V10/V11 REORG Enhancements
REORG Enhancements in DB2 10

- REORG SHRLEVEL CHANGE for entire catalog & directory
- REORG SHRLEVEL CHANGE for LOB table spaces
- Additional RTS information for intelligent REORG
- Partition-level REORG performance with NPSIs due to index leaf page list prefetch
- REORG INDEX performance due to index leaf page list prefetch
- REORG REBALANCE on partitioned tables with LOB columns
- ALTER LIMITKEY support for partitioned tables with LOB columns
- Ability to use SHRLEVEL REFERENCE or CHANGE to remove REORP state after alter of limitkey values
Automated mapping table handling in DB2 V11

• Support mapping tables in PBG
  – Increases mapping index limit from 64Gb to 16Tb
  – Retrofitted to V9 in PM58177

• Mapping table DDL must change in 11 due to RBA/LRSN change

• Requirements to automate mapping tables

• So… New automated mapping tables in REORG
  – Automatically create new format mapping table if required
    • If mapping table specified & correct format then honor specification
    • Else if specified but incorrect format then create new in same db as original
    • Else if not specified and zparm DB specified then create in zparm DB
    • Else create in implicitly created DB
    • DROP at end of REORG or end of last REORG if multiple REORGs in job step
  – NFM requires new format mapping table, CM supports both old and new
  – No additional auth requirements necessary for creation of mapping tables
REORG without sorting data

- Increasingly REORGs are performed for reasons other than to regain clustering of data, yet no ability to avoid cost of reclustering
- Prior to DB2 11, REORG SHRLEVEL CHANGE did not support SORTDATA NO
- DB2 11 supports SORTDATA NO with SHRLEVEL CHANGE
- New RECLUSTER YES/NO option on SORTDATA NO
  - RECLUSTER NO – Do not unload data through clustering index and do not sort data records in clustering order
REBALANCE enhancements

• Improved availability & failure prevention

• Support REORG SHRLEVEL CHANGE REBALANCE
  – Complements online ALTER of partition limit keys

• Improve resiliency with enhanced distribution algorithm & improved handling of empty partitions

• Build compression dictionary for all partitions
  – Previously, partitions that were empty at the start of REORG would not have a dictionary built, requiring a subsequent REORG to gain compression

• New SORTCLUSTER option to sort data in clustering as well as partitioning order to avoid AREO*
  – Occurred when partitioning key not a superset of clustering key
  – SORTCLUSTER YES – sort in partitioning and clustering order, avoid AREO*
  – No specification – keep existing behaviour, set AREO*
  – SORTCLUSTER NO – keep existing behaviour, but do not set AREO*
Partition-level inline image copy

• Faster partition-level recovery from inline image copy
• Create partition-level inline image copies if using TEMPLATE with &PA or &PART
  – No new option or keyword on REORG
  – PM93611:
    • Support substring notation with &PA as long as substring ensures uniqueness
    • Support writing to tape as long as STACK YES not specified
  – Example of a template used by COPYDDN:
    – TEMPLATE SCOPY1 UNIT(SYSDA) DISP(MOD,CATLG,CATLG)

• RECOVER of single partition of a 20 partition table space
  – ET reduced by 28%
  – CPU reduced by 49%
REORG of PBGs

Smarter software for a smarter planet

Information Management
REORG of PBG

- No REORG parallelism
- REORG of single partition must fit rows back into that partition
- REORG of multiple parts can redistribute rows across partitions
  - Will fill up earlier partitions first
- Prior to V10, REORG of base with LOB columns cannot redistribute rows across partitions
- Empty partitions will not be deleted
- REORG cannot grow new parts unless REORG at table space level
- REORG at table space level will grow new parts as needed
  - Not prior to V10 if LOB columns exist
  - REORG of single part or subset of parts will not grow new parts
  - New LOB table spaces will be added, but will be left copy-pending
REORG of PBG

- **REORG fails if rows must fit back into partition but cannot**
  - Possible causes:
    - PCTFREE & FREEPAGE
    - Change in compression ratio or alter to COMPRESS NO
    - BRF-RRF conversion
    - MAXROWS change
    - Change to MEMBER CLUSTER
    - Etc

- **What to do if REORG fails because rows won’t fit?**
  - Use SHRLEVEL CHANGE to avoid application impact
  - View as single table and REORG whole PBG table space
  - Use REORG_IGNORE_FREESPACE zparm on part-level PBG REORG
    - Introduced by PK83397
  - If LOB columns exist then may need to UNLOAD/RELOAD if pre-V10

- **Prior to DB2 10 be careful about using PBG for tables with LOB columns**
Physically delete empty PBG partitions

- Ability for REORG to physically delete empty PBG partitions in DB2 V11
- New zparm REORG_DROP_PBG_PARTS
  - DISABLE – keep V10 behavior (default)
  - ENABLE – Delete empty PBG partitions on table space-level REORG
- Considerations:
  - Cannot be specified on REORG statement
  - If PBG created using NUMPARTS or ALTER ADD partition used, REORG may prune to a lesser number of partitions
  - No PIT recovery to prior to a pruning REORG
    - No facility to resurrect deleted partitions
REORG of LOBs

Smarter software for a smarter planet

Information Management
REORG & LOBs

- **REORG of LOB table spaces**
  - V6 – V8
    - In-place REORG
    - SHRLEVEL NONE & LOG YES only
    - Incomplete rechunking of LOBs
    - No reclamation of physical space
    - No sort required
  - V9
    - SHRLEVEL REFERENCE
    - No logging
    - Perfect rechunking of LOB data
    - Reclaims physical space
    - No sort required
  - V10
    - SHRLEVEL CHANGE
    - No mapping table required
    - SHRLEVEL NONE deprecated in 10 NFM – will complete RC0 but no action is taken
    - Convert REORG jobs to SHRLEVEL REFERENCE or CHANGE before 10 NFM
REORG & LOBs

- New AUX keyword on REORG of partitioned base for improved LOB handling in DB2 10
  - Permit rows to flow between partitions
  - Supports REORG REBALANCE with LOB columns
  - Supports ALTER of LIMITKEY with LOB columns
  - Permits move of rows between parts on PBG REORG
  - Permits deletion of corresponding LOBs on REORG DISCARD
  - Default is AUX NO unless LOB objects required to complete REORG
  - No XML column support for classic partitioned or PBR
  - No mapping table change
What and when to REORG
REORG avoidance

• Why REORG if you don’t have to?
• Use DSNACCOR/DSNACCOX, own application logic or automation tool to determine what to REORG and when
• If REORG purely to gain compression
  • Consider LOAD COPYDICTIONARY instead
  • DB2 10 builds compression dictionary on the fly on insert
• New REORGCLUSTERSENS RTS column in V10
  • Do not REORG to regain clustering if no application benefit
• Index leaf page list prefetch in V10 can reduce need to reorg indexes
REORG INDEX vs. REBUILD INDEX

- REORG is not a substitute for REBUILD, but REBUILD could be a substitute for REORG
- REBUILD INDEX SHRLEVEL CHANGE provided in V9
  - Excellent for create of new non-unique indexes and for indexes that are broken or already in RBDP
  - Does not operate against a shadow, so will set RBDP if not already set
  - Note SHRLEVEL CHANGE means data is RW, not index – index is RBDP
- REORG INDEX operates against a shadow – better availability than REBUILD INDEX
- REBUILD can be faster in V9, particularly if index is disorganized
- REORG INDEX performance improvement in V10 due to prefetch
- Summary:
  - If you need to REBUILD then REBUILD
  - If you need to REORG on V9, and you can tolerate the index being RBDP, then REBUILD may be quicker
  - If you need to REORG on V10 then REORG
Recommendations, Best Practices & Summary
Recommendations & Best Practices

- Stay current on maintenance & DB2 releases and take advantage of new features
- Reorg multiple partitions in a single REORG statement, particularly if NPSIs exist
- If reorging many parts, and NPSIs exist, consider whether REORG of entire table space is quicker
- If NPSIs disorganised, move to V10 for faster REORG INDEX & part-level REORG
- Use DSNACCOX, Automation Tool or other logic to only reorg what needs to be reorged
- Use SORTNUM elimination (UTSORTAL=YES, IGNSORTN=YES)
- Consider REBUILD INDEX instead of REORG INDEX if pre-V10 and index is disorganised
Recommendations & Best Practices

- REORG SHRLEVEL CHANGE for maximum availability
- Use DRAIN ALL rather than DRAIN WRITERS
- Use TIMEOUT TERM to free up objects on timeouts
- If minimising application impact is key:
  - \((\text{DRAIN\_WAIT} + \text{MAXRO}) < (\text{IRLMRWT} -5 \text{ or } 10 \text{ secs})\) for minimal application impact
  - Specify high RETRY value (6 or more)
- If REORG success in a small window is key:
  - Consider starting REORG early with MAXRO DEFER then -ALTER UTILITY command
  - High DRAIN\_WAIT & MAXRO to guarantee REORG success
- If using REORG DISCARD, use NOPAD for improved performance
- LOBs
  - SHRLEVEL REFERENCE in V9, SHRLEVEL CHANGE in V10
  - Stop using SHRLEVEL NONE before DB2 10 NFM
Summary

• The REORG utility is essential for data management and maintaining application performance

• As data volumes grow, it is critical that customers have confidence in the ability to manage that data

• IBM will continue to deliver availability, performance and functional enhancements to meet these needs