



# IBM DB2 for z/OS Utility Update: Best Practices

Bryan F. Smith [bfsmith@us.ibm.com](mailto:bfsmith@us.ibm.com)  
IBM

August 27, 2009  
6140



maxi  
ove agility save tim  
enges colleagues



# Agenda

- Availability
- Performance
- Features & function
- Best practices
- Summary

# Availability – what has changed recently?

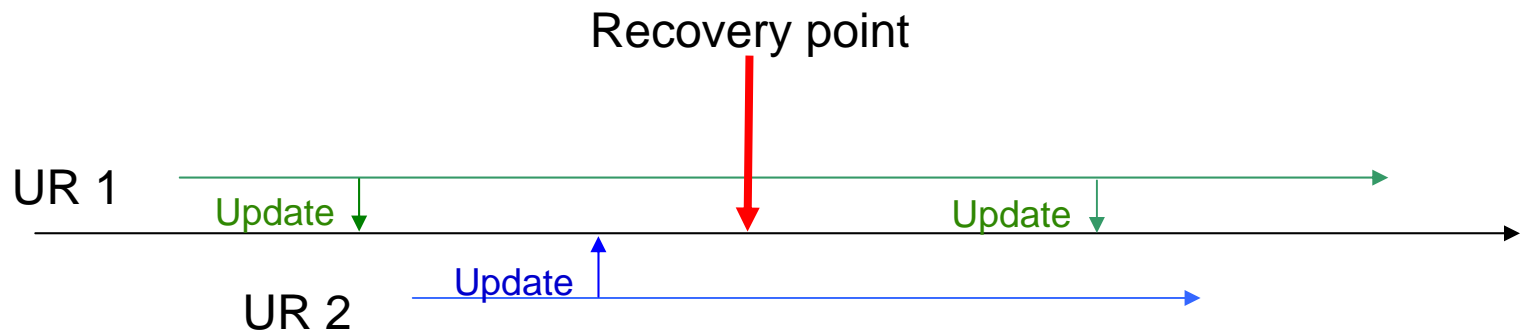


- Online create or rebuild of non-unique indexes
  - REBUILD INDEX SHRLEVEL CHANGE
- Eliminate outage for partition-level REORGs
  - Eliminate BUILD2 phase
- Avoid need for REORG to get compressed data
  - LOAD COPYDICTIONARY
  - PK63324 & PK63325 (V9)
- Online data consistency checking and repair
  - CHECK DATA SHRLEVEL CHANGE
  - CHECK LOB SHRLEVEL CHANGE
  - REPAIR LOCATE... SHRLEVEL CHANGE
- Run data consistency checks without impacting BACKUP SYSTEM or disk mirroring
  - PK41711 (V9)

# Availability – what has changed recently?



- Replace data with virtually no outage
  - CLONEs effectively provide LOAD REPLACE SHRLEVEL CHANGE
  - UTS only
- Read LOB data during REORG
  - REORG SHRLEVEL REFERENCE for LOBs
- RECOVER to point in time with consistency
  - Avoid need for QUIESCEs



# Performance – what has changed recently?



- Faster REORGs
  - Parallel unload of partitions
  - Parallel reload of partitions
  - Parallel log apply
    - Greater likelihood of REORG keeping up with logging rates
- Faster CHECK INDEX SHRLEVEL REFERENCE
  - Parallel index processing
- Up to 40% faster COPY & RECOVER RESTORE phase to/from tape
  - Support Large Block Interface for image copies to tape
- Reduced impact on applications when running COPY
  - COPY uses MRU for buffers to improve BP hit ratio for online applications
- Reduced impact on applications when running LOAD & REORG
  - Auto-invalidate of cached dynamic statements on completion of LOAD & REORG
  - PK47083 (V8 & V9)

# Performance – what has changed recently?



- Greater utility parallelism with SORTNUM elimination
  - PK45916 (V8), PK41899 (V9)
  - Major improvement in utility sort processing
  - Simpler, more efficient, more reliant on RTS
- SORTBLD performance improvement
  - PK60956 (V8 & V9)
  - Up to 20X performance improvement in SORTBLD for indexes with small SECQTY
- LOAD & REORG performance improvement
  - PK61759 (V8 & V9)
  - 10% CPU & elapsed time improvement in RELOAD phase
  - 10% CPU reduction in SORT phase
- COPY performance improvement
  - PK74993 (V9)
  - 20% elapsed time improvement for copy of multiple small datasets to tape
- COPY performance with large LISTDEF lists
  - PK78865 (V8 & V9)
  - Reduce writes to SYSUTILX

# Performance – what has changed recently?



- Crossloader performance improvement for CCSID data conversion
  - PK76860 (V8 & V9)
- LOAD/UNLOAD LOB file reference variable performance
  - PK75216 (V9)
  - PDS only, not HFS
- UNLOAD performance for multi-table table spaces
  - UTILINIT phase – use DBD rather than catalog lookup
  - PK77313 (V8 & V9)
- REORG PART of empty partition performance
  - Avoid NPI scan for non-clustering indexes
  - PK67154 (V8 & V9)

# Performance – what has changed recently?



- LOAD and UNLOAD to/from virtual file
  - USS named pipe support with templates
  - PK70269 (V8 & V9)
- DSN1COPY performance
  - Improved VSAM buffer allocation for page sets with cylinder allocation
  - PK78516 (V8 & V9)
- RUNSTATS histogram statistics
  - Improved query optimization for non-uniform distribution
  - Example - 1, 3, 3, 4, 4, 6, 7, 8, 9, 10, 12, 15 (sequenced), cut into 3 quantiles

Seq No	Low Value	High Value	Cardinality	Frequency
1	1	4	3	5/12
2	6	9	4	4/12
3	10	15	3	3/12

# Performance – what has changed recently?



- CPU cost reduction in V9
  - 10-20% for COPY & RECOVER
  - 5-30% for LOAD, REORG, REBUILD INDEX
  - 20-60% for CHECK INDEX
  - 35% for LOAD partition
  - 30-40% for RUNSTATS INDEX
  - 40-50% for REORG INDEX
  - 70% for LOAD REPLACE partition with dummy input
- zIIP enablement for utility index processing in V8

## What's New in DB2 for z/OS Utilities and zIIP?

- Additional zIIP engine offload for the DB2 Utilities
  - For DB2 Version 8 and DB2 9
- What is eligible?
  - Almost all DB2 utilities sorting of fixed-length records in the memory object sort path
- System requirements
  - System z9 or z10 with specialty engines and z/OS V1.10 with PTF UK48846 from APAR PK85856 and PTFs for either DB2 Version 8 or DB2 9
  - For DB2 Version 8:
    - PTF UK48911 from APAR PK85889
  - For DB2 :
    - PTF UK48912 from APAR PK85889

## Customer Value

- Almost all utility sort processing involves fixed-length records, so this new support will apply to most utilities
  - Exception is REORG data sorts that handle variable length records and is not eligible for the offload
- Most utility processing involves sorting index keys
  - Can represent a significant amount of data
  - Can consume up to 60% of utility CPU time in sort processing
- Initial testing determined approximately **50%\*** of sort CPU time will be offloaded with this enhancement depending on
  - if the sort processing selects the memory object sort path
  - the size of data being sorted
  - available system resources
- Who will benefit?
  - Customers who are constrained on CPU during utility processing or who would like to reduce internal CPU charge back costs for the utilities

## Where can I find more information?

- IDUG Conference in Rome on October 5-9<sup>th</sup>
  - Session A17: “What’s New in DB2 for z/OS Utilities”
- Information on Demand (IOD) Conference in Las Vegas on October 25-29
  - Session 1608: “Maximizing the Performance of DB2 9, Db2 Utilities and System z10”
  - Session 1456: “DB2 Utilities Update!”
- <http://www-01.ibm.com/software/data/db2imstools/solutions/utilities-mgmt.html>

# Features & function – what has changed recently?

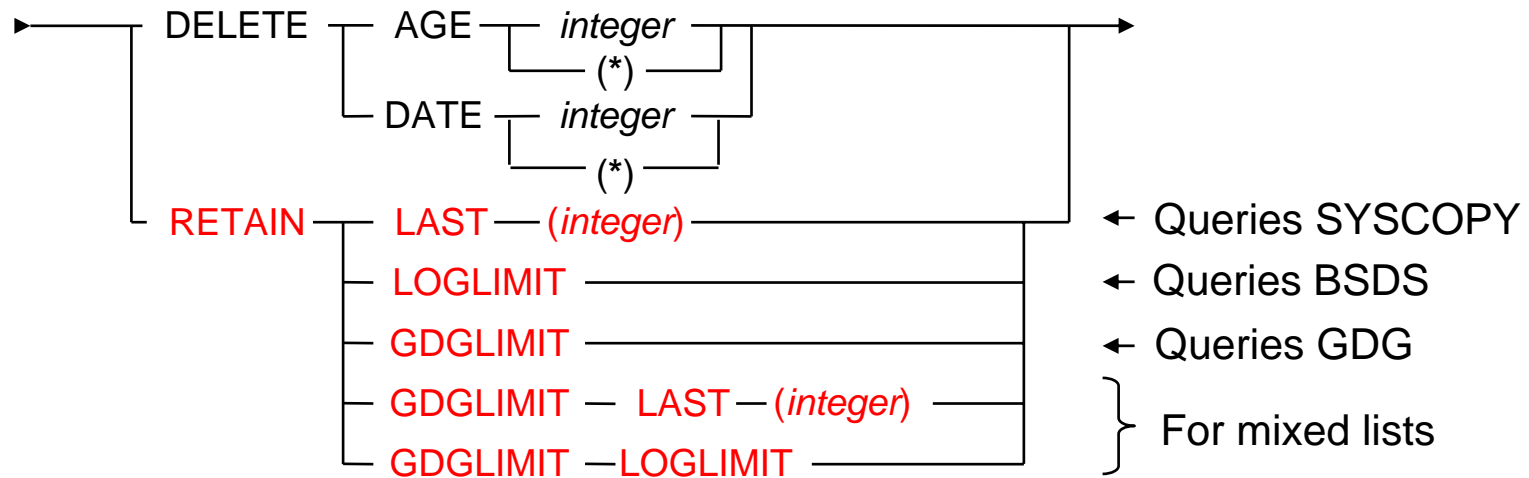


- BACKUP SYSTEM & RESTORE SYSTEM enhancements
  - Support for tape
  - Support for incremental FlashCopy
- Object-level recovery from system-level backup
- RECOVER to any point in time with consistency
- SORTNUM elimination
  - Simplified utility invocation
- Remove restriction on REORG of >254 compressed parts
  - ZPARM restricts LOAD in V9 – restriction removed in DB2 X
- Better information for DPRPR/QRep or other IFI 306 readers
  - Write diag log record at utility termination so IFCID 306 readers can trigger refresh
  - PK78558 (V9)

# Features & function – what has changed recently?



- MODIFY RECOVERY simplification & safety



- Template switching for COPY utility
  - E.g. copy to disk if small, to tape if large

```

TEMPLATE LRG DSN &DB..&TS..D&DA..T&TI. UNIT=TAPE
TEMPLATE SML DSN &DB..&TS..D&DA..T&TI. UNIT=SYSALLDA LIMIT(20 CYL, LRG)
COPY TABLESPACE SMALL.TS COPYDDN(SML)
COPY TABLESPACE LARGE.TS COPYDDN(SML)
  
```

# Features & function – what has changed recently?



- Permit use of ALIASes for LOAD, RUNSTATS and UNLOAD
  - PK77061 (V9)
- New DSNACCOX stored procedure to gather statistics from catalog and make utility recommendations
  - See PK44133
  - DSNACCOR still supported
- More information
  - All utility messages in job output have julian date & timestamp
  - -DISPLAY UTILITY enhanced to show progress of logapply

```
DSNU116I csect-name RECOVER LOGAPPLY PHASE DETAILS:  
STARTING TIME = timestamp  
START RBA = ss START LRSN = rr  
END RBA = ee END LRSN = nn  
LAST COMMITTED RBA = cc LAST COMMITTED LRSN = ll  
ELAPSED TIME = hh:mm:ss
```

# What's coming?

- Remove usability restrictions for REORG
  - LOBs, PBG, catalog/directory, rebalancing,...
- REORG avoidance
- Remove UTSERIAL lock for greater utility concurrency
- RTS enhancements & greater reliance upon RTS
- Intelligent & autonomic statistics gathering
- BACKUP SYSTEM / RESTORE SYSTEM enhancements
- FlashCopy exploitation
- LOAD & UNLOAD enhancements
  - Improved LOB/XML processing
  - Improved UTF-16 support
- CHECK utility enhancements
  - XML, availability, data correction,...
- Faster point in time recovery
- Faster & better COPY processing
  - Incremental, CHANGELIMIT, FlashCopy

# COPY Best Practices

- COPY
  - PARALLEL keyword provides parallelism for lists of objects (including partitions)
  - CHECKPAGE YES incorporated into V9 - look for RC=8!
  - Maximize other utilities' access to objects while copying a list with SHRLEVEL CHANGE and OPTIONS EVENT(ITEMERROR,SKIP)
    - Keeps objects in the list in UTRW state \*only\* as each object is being copied instead of for the duration of the COPY utility
    - UTRW – utility allows read/write access by applications, but no access for exclusive utilities
  - Incremental copy rule-of-thumb: Consider using incremental image copy if
    - <5% of pages are randomly updated (typically means less than 1% of rows updated)
    - <80% of pages are sequentially updated
    - Incremental image copies use list prefetch, so monitor for rid list pool full conditions
  - Copy indexes on your most critical tables to speed up recovery
- MERGECOPY – consider using it

# RECOVER/QUIESCE Best Practices

- RECOVER
  - PARALLEL keyword provides parallelism for lists of objects (including partitions)
  - Compressed pagesets result in faster restore phase
  - Enable Fast Log Apply (which can use dual-copy logs) and PAV
    - $\leq 10$  jobs/member with LOGAPSTG=100MB, up to 99 objects per RECOVER
  - For recovery to a prior point in time
    - Always recover related sets of objects together (same RECOVER utility statement)
  - DB2 9 for z/OS: recover to PIT with consistency
    - Backs out uncommitted changes for the objects specified on the RECOVER utility statement
    - Significantly reduces the need to run QUIESCE, which can be disruptive to applications
- QUIESCE
  - WRITE NO is less disruptive (no quiescing of COPY=NO indexes)
  - Use TABLESPACESET
  - Do you still need it in V9?

# MODIFY RECOVERY Best Practices

- Base your MODIFY strategy on your backup strategy and not vice versa
- REORG SYSLGRNX regularly
- Run MODIFY RECOVERY regularly to clean up old records in SYSCOPY and SYSLGRNX
- DB2 9 has RETAIN LAST n, GDGLIMIT and BSDS options
- Also resets “ALTER\_ADD\_COLUMN” flag in OBD when deleting image copies with previous row versions
  - MODIFY RECOVERY DELETE AGE/DATE to delete everything before the REORG that follows the ALTER
  - Will make next REORG more efficient if no more old row versions exist
- Remember that MODIFY RECOVERY works on day boundaries

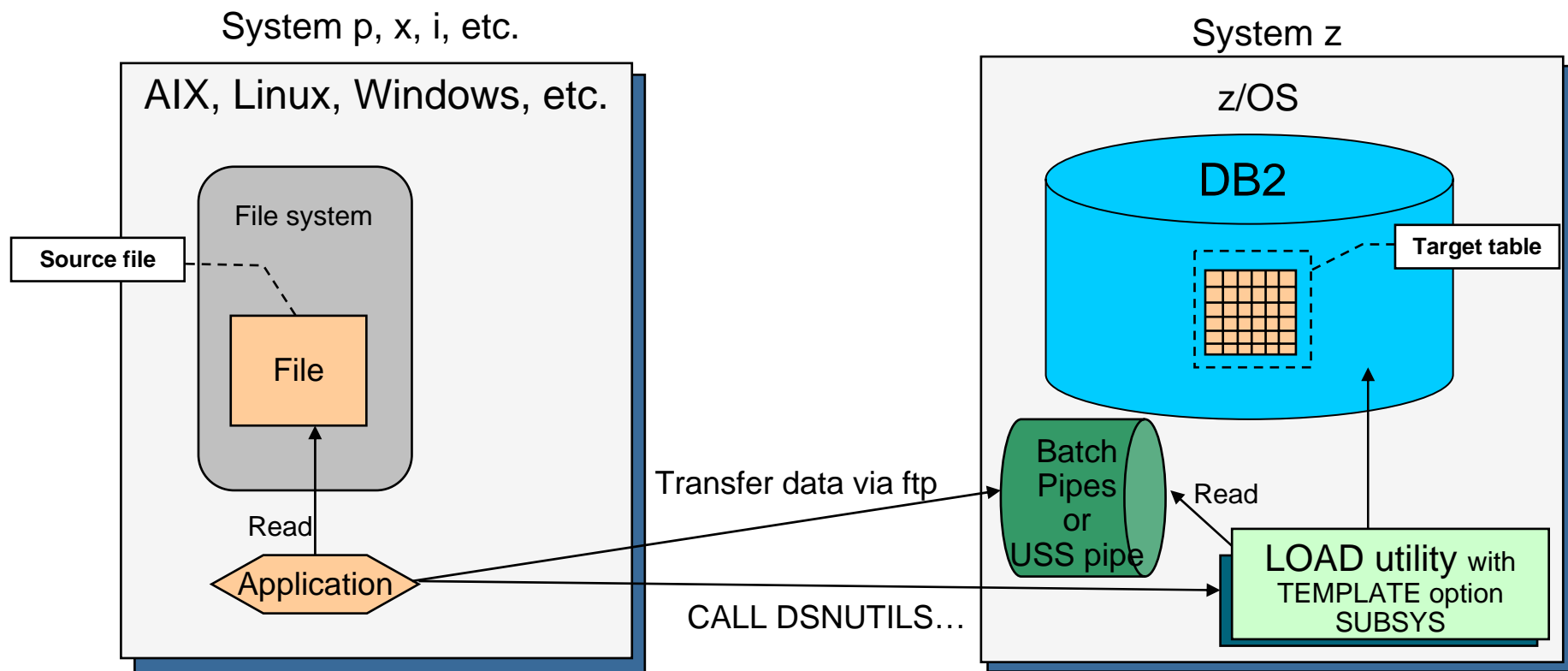
# LOAD Best Practices

- LOAD
  - LOG NO reduces log volume; if REPLACE, then take inline copy
  - KEEPDICTIONARY (track dictionary effectiveness with history statistics PAGESAVE) - small performance impact if loading lots of data
  - 254 partition limit for compressed table spaces can be lifted by DBA
    - PK51853 shipped new ZPARM MAX\_UTIL\_PARTS (watch virtual storage)
  - Load Partition Parallelism (V7)
    - Not individual LOAD part level jobs
    - Enable Parallel Access Volume (PAV)
  - Index parallelism (SORTKEYS)
    - Provide value for SORTKEYS when input is tape/PDS mbr or variable length
    - SORTKEYS is the sum of ALL indexes (and foreign keys) on the table
    - Remove SORTWKxx / UTPRINxx, and turn on UTSORTAL=YES

## LOAD Best Practices contd.

- LOAD
  - Inline COPY & Inline STATISTICS
  - Use REUSE to logically reset and reuse DB2-managed data sets without deleting and redefining them (affects elapsed time)
  - When using DISCARD, try to avoid having the input on tape
    - Input is re-read to discard the errant records
  - Sort data in clustering order (unless data is randomly accessed via SQL)
  - LOAD RESUME SHRLEVEL CHANGE instead of batch inserts
  - “LOAD REPLACE SHRLEVEL CHANGE” can be achieved by loading into clone table and then exchanging the tables on DB2 9
  - LOAD via Batchpipes or USS pipes to load data that is transferred via FTP from clients – see PK70269

# LOAD/UNLOAD via Batchpipes or USS pipes



# REORG Best Practices

- REORG
  - Use SHRLEVEL REFERENCE or SHRLEVEL CHANGE
  - Inline COPY & Inline STATISTICS
  - KEEPDICTIONARY (track dictionary effectiveness with history statistics PAGESAVE) – large performance impact
  - 254 partition limit for compressed table spaces in V8
    - PK51853 shipped new ZPARM MAX\_UTIL\_PARTS (watch virtual storage)
    - DB2 9 for z/OS no longer has this limit and uses virtual storage more effectively
  - Remove SORTWKxx / UTPRINxx, and turn on UTSORTAL=YES
  - Run REORG against as many partitions as possible in the same job or against the whole table space

# REORG Best Practices contd.

- REORG
  - Partition parallelism in DB2 9 and NPI processing
    - Parallel REORG jobs for same table space but different partitions no longer supported if NPIs defined
    - After REORG PART with no BUILD2 phase, less need for REORG NPI
    - Watch out for LISTDEFS at partition level with NPIs - full REORG might be more efficient
  - SHRLEVEL NONE if constrained for disk space
    - LOG NO reduces log volume; requires an image copy (inline is a good choice)
    - NOSYSREC to avoid I/O (forced for SHRLEVEL CHANGE)
      - *Take full image copy before REORG SHRLEVEL NONE*
    - Use REUSE to logically reset and reuse DB2-managed data sets without deleting and redefining them (improves elapsed time)

## REORG Best Practices contd.

- REORG
  - SORTDATA NO only if data is already in or near perfect clustering order and disk space is an issue
  - Set appropriate PRIQTY/SECQTY to minimize extend processing
    - PK60956 helps to improve SORTBLD elapsed time up to 20x for indexes with small SECQTY!!!
    - SORTBLD elapsed up to 20x improvement!!!
    - Affects all utilities that are (re-)building indexes
  - Run MODIFY RECOVERY some time after ALTER TABLE ... ADD COLUMN

## REORG Best Practices contd.

- REORG SHRLEVEL CHANGE (sometimes called online REORG)
  - TIMEOUT TERM frees up the objects if timeouts occur in getting drains
  - DRAIN ALL (better chance of entering SWITCH phase)
  - $(\text{DRAIN\_WAIT} + \text{MAXRO}) < (\text{IRLMRWT} - 5 \text{ or } 10 \text{ seconds})$ 
    - Avoid application timeouts
    - But don't set MAXRO too low
  - RETRY = utility lock timeout multiplier (6 by default)
  - $\text{RETRY\_DELAY} = \text{DRAIN\_WAIT} * \text{RETRY}$
  - Enable detection of long running readers (zparm) and activate IFCID 0313 (it's included in STATS CLASS(3))
    - This will report readers that may block command and utilities from draining
    - It includes "well-behaved" WITH HOLD cursors which a drain cannot break-in on
  - More Joys of Commitment by Bonnie Baker
    - [http://www.db2mag.com/db\\_area/archives/2003/q1/programmers.shtml](http://www.db2mag.com/db_area/archives/2003/q1/programmers.shtml)

## REORG Best Practices contd.

- REORG SHRLEVEL CHANGE
  - Consider scheduling SWITCH phase in a maintenance window to avoid concurrent workloads that may prevent the utility from breaking in:
    - MAXRO DEFER and LONGLOG CONTINUE will let REORG do its job except for the last log iteration and the switching
    - REORG will continue applying log until MAXRO is changed with the ALTER UTILITY command
    - Many log iterations might reduce the “perfect” organization of the table space, so keep the time until MAXRO is changed to allow final processing down to a minimum

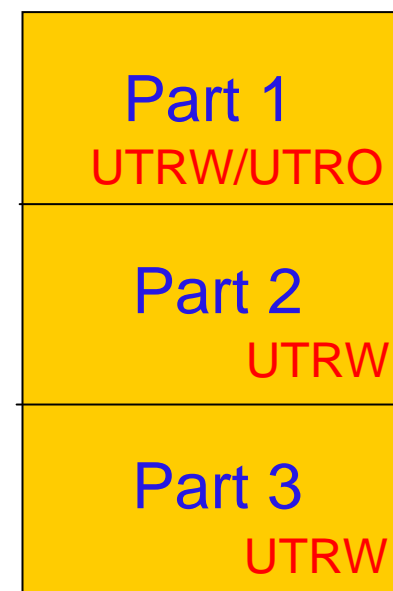
## REORG LOB Best Practices

- DB2 V8 only REORG LOB SHRLEVEL NONE if performance degraded because of bad LOB chunking
- DB2 9 - use SHRLEVEL REFERENCE
  - Available in CM
  - Reclamation of unused space
  - Full read access to LOBs except during SWITCH phase
  - Inline imagecopy required to maintain recoverability
  - No restart capability
    - Shadow pageset discarded in event of failure
  - SHRLEVEL NONE still supported
    - Remains default, but will be deprecated in future

## A word about PBGs

- No LOAD/REORG parallelism
- No pruning of partitions in V9
- No load at partition level
- REORG of single part
  - No new part creation
  - Rows must fit back into part, but may not!
- REORG of part range
  - Data can flow from one part to another within range
  - If LOB column exists then rows will not move between parts in V9
- Recommendation:
  - View as single table and REORG as a whole

### PBG



# REBUILD INDEX Best Practices



- REBUILD INDEX
  - Indexes are built in parallel
  - Remove SORTWKxx / UTPRINxx and use SORTDEVT/SORTNUM or UTSORTAL=YES
  - Inline STATISTICS
  - Use REORG INDEX SHRLEVEL CHANGE to move index data sets to different volumes
  - CREATE INDEX DEFER followed by REBUILD INDEX
    - As of V8, dynamic SQL will not select the index until it is built
  - DB2 9 allows SHRLEVEL CHANGE
    - Unique indexes are put in RBDP because uniqueness can not be checked during rebuild process, so no INSERT/UPDATE allowed that affects unique index
    - No parallel jobs on different indexes of the same table space -> use single job with multiple indexes specified

# Common Sort Related Problems in DB2 for z/OS Utilities



- SORTNUM - “One size doesn’t fit all”
  - Many sorts in the same utility invocation have very different needs for sort work data sets
  - Utilities allow only a single SORTNUM specification which tells DFSORT into how many data sets it should split the sort work space
  - Large sorts require many data sets to satisfy the need but the same number will then also be applied to the smaller sorts allocating valuable resources
  - DASD situation varies: SORTNUM 4 might work today, but tomorrow even SORTNUM 8 might fail
- Large number of data sets will limit possible degree of parallelism as each data set requires storage (mainly below the line)
- Sort work space overhead caused by different key lengths when sorting multiple indexes in the in the same sort task
- Estimates for sort work space sometimes too high

## “SORTNUM Elimination” Concept

- DB2 determines sort work data set size for each DFSORT invocation using information from Real-Time Statistics
  - DB2 allocates sort work data sets before invoking DFSORT
    - Ensure that disk space can be allocated
    - Know exactly how many data sets were allocated
    - Determine degree of parallelism according to storage consumption
  - Invoke DFSORT with pre-allocated sort work data sets
  - DB2 de-allocates data sets after sort completion
- Developed in close cooperation with several customers from Germany, USA, Netherlands*



# **SORTNUM Elimination Details**

- Required sort work space is divided by two for first allocation
  - Maximum data set size used is Mod-54 volume size
- Large data sets are supported, consider using large sort volumes
- If allocation fails, size is gradually reduced until allocation is successful
  - You may see some DFSMS failure messages, e.g. IDG17272
- Allocation of further data sets continues with previous size until total required space has been satisfied
  - Number of data sets varies even for same job depending on current DASD situation
- Utilities with parallelism
  - Data sets with largest space requirements are allocated first
  - Use actual number of allocated data sets for calculation of possible degree of parallelism

# SORTNUM Elimination Externals

- New function is controlled by two new system parameters:
  - **UTSORTAL**: Utility Sort Allocation
    - Default is NO, online changeable, member scope
    - If YES then
      - *DB2 allocates sort work data sets as long as SORTNUM option was NOT used*
      - *RTS will be used for estimates when available*
  - **IGNSORTN**: Ignore SORTNUM
    - Default is NO, online changeable, member scope, only applicable if UTSORTAL=YES
    - If YES then all SORTNUM specifications in utility statements are ignored
    - If NO then existing SORTNUM specifications are honored and DFSORT will do the dynamic allocation of sort work data sets just as before

# **SORTNUM Elimination External (contd.)**

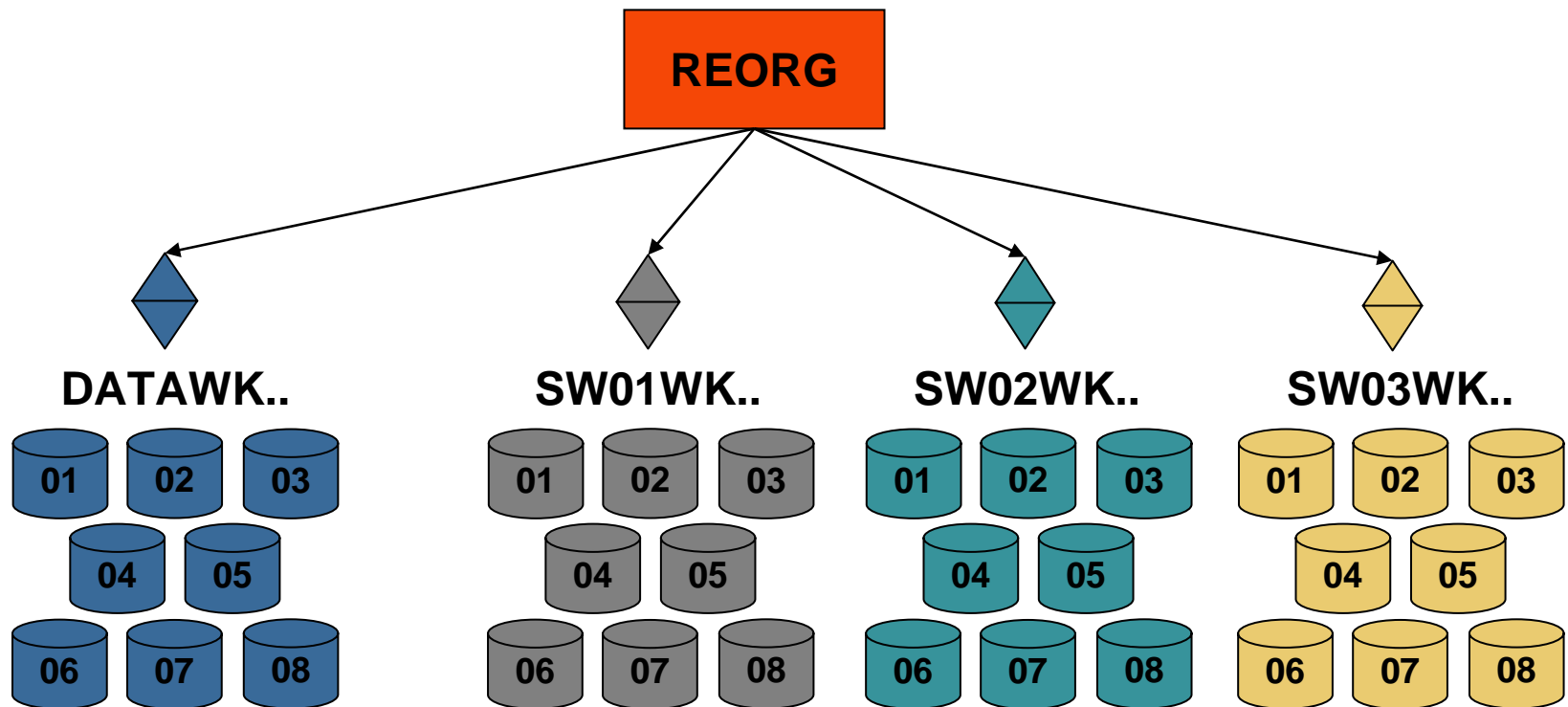


- Hard coded sort work DD cards always disable DB2 allocation of sort work data sets
- Message “DSNU3340I - UTILITY PERFORMS DYNAMIC ALLOCATION OF SORT DISK SPACE” indicates use
- Used for all sorts in: LOAD, REORG, CHECK INDEX, REBUILD INDEX, CHECK DATA, RUNSTATS

# Sample 1: Sort Work Data Sets allocated by DFSORT



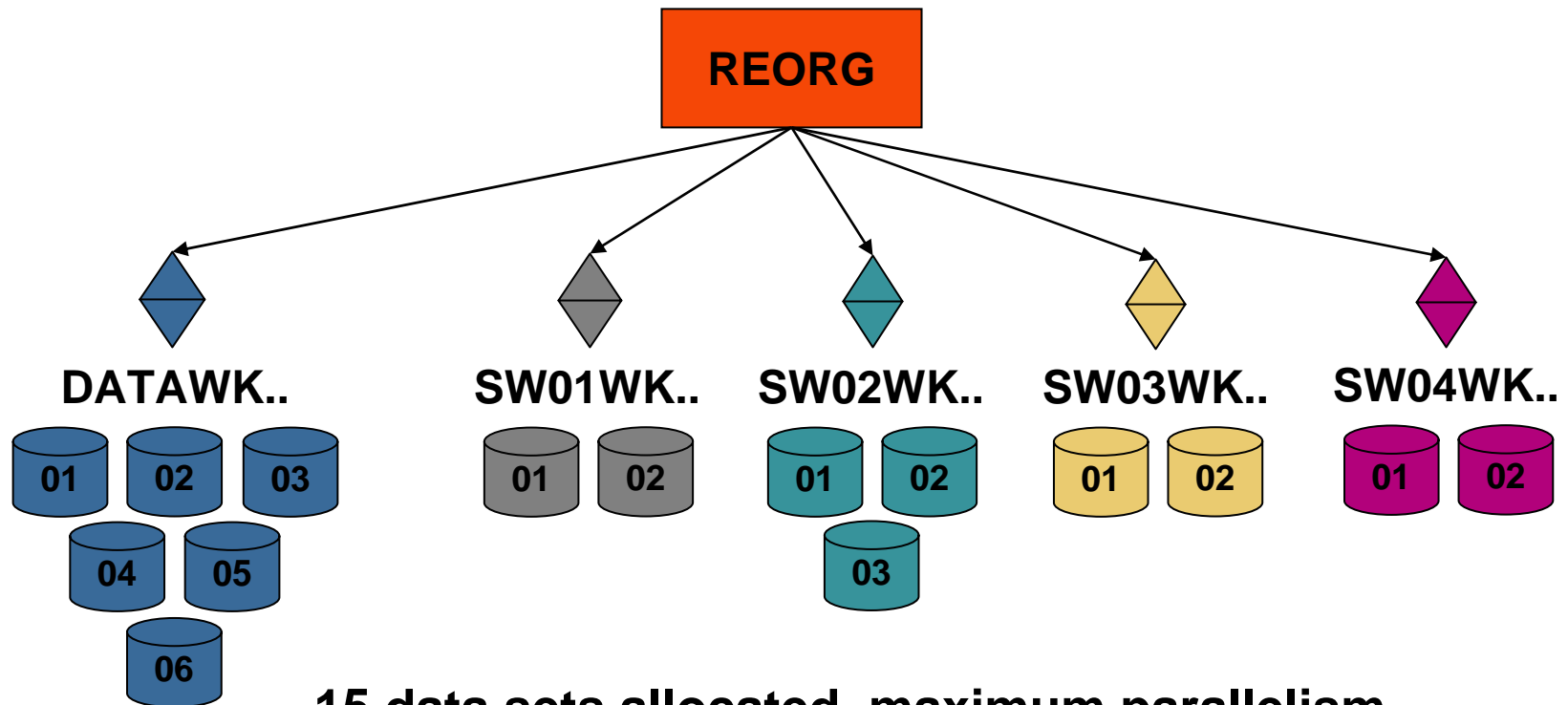
- Sample: REORG TABLESPACE ... SORTNUM 8 with 4 indexes, index build parallelism constrained to 3 sort tasks



**32 data sets allocated, parallelism constrained**

# Sample 2: Sort Work Data Sets allocated by DB2

- Sample: REORG TABLESPACE with 4 indexes and UTSORTAL=YES, index build parallelism with 4 sort tasks

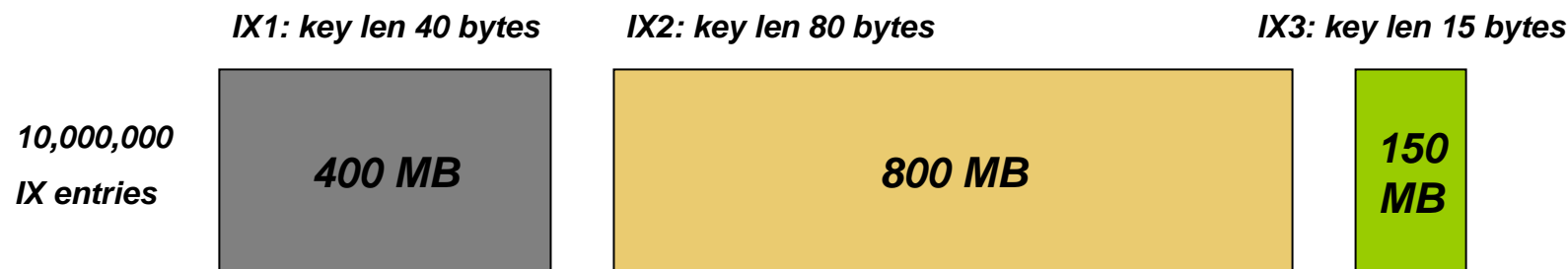


# Dealing with Parallel Sort Tasks

- Many of our utilities offer parallel sort tasks to reduce elapsed time
- Ideal number of parallel tasks is either the number of indexes or the number of partitions
- In constrained systems the number of parallel tasks will be lower than that
- Multiple “logical” sort tasks have to be combined into the same “physical” sort task
- If key length does not match, the resulting sort task has to deal with overhead in the amount of data being sorted and thus the sort work data sets that need to be allocated grow more than necessary just for the purpose of sorting
- When multiple indexes have to be sorted in the same subtask due to constraints, index assignment is optimized to minimize the wasted disk space
  - For LOAD, REORG V8, REBUILD INDEX, CHECK INDEX
  - Only done when DB2 allocated the sort work data sets

# Sample: Multiple Sort Tasks

- Optimal distribution: 3 tasks for 3 indexes

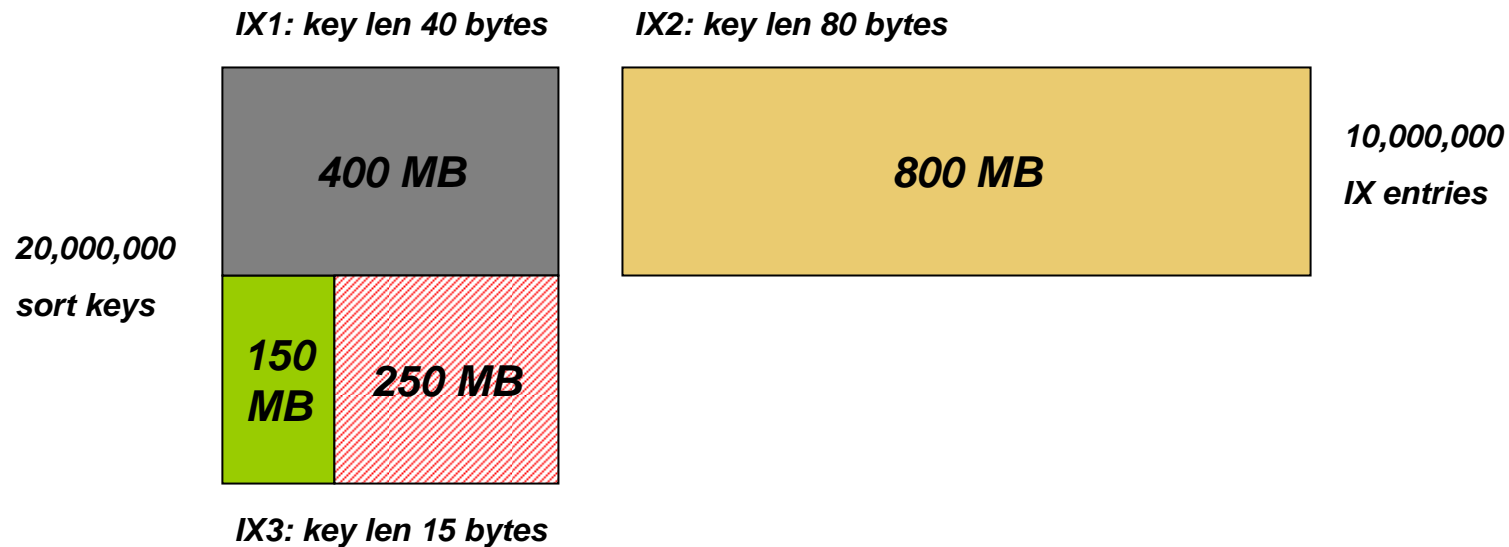


Required space: 1,350 MB

# Sample: Combining Multiple Sort Tasks (1)



- Constrained system: 2 tasks for 3 indexes

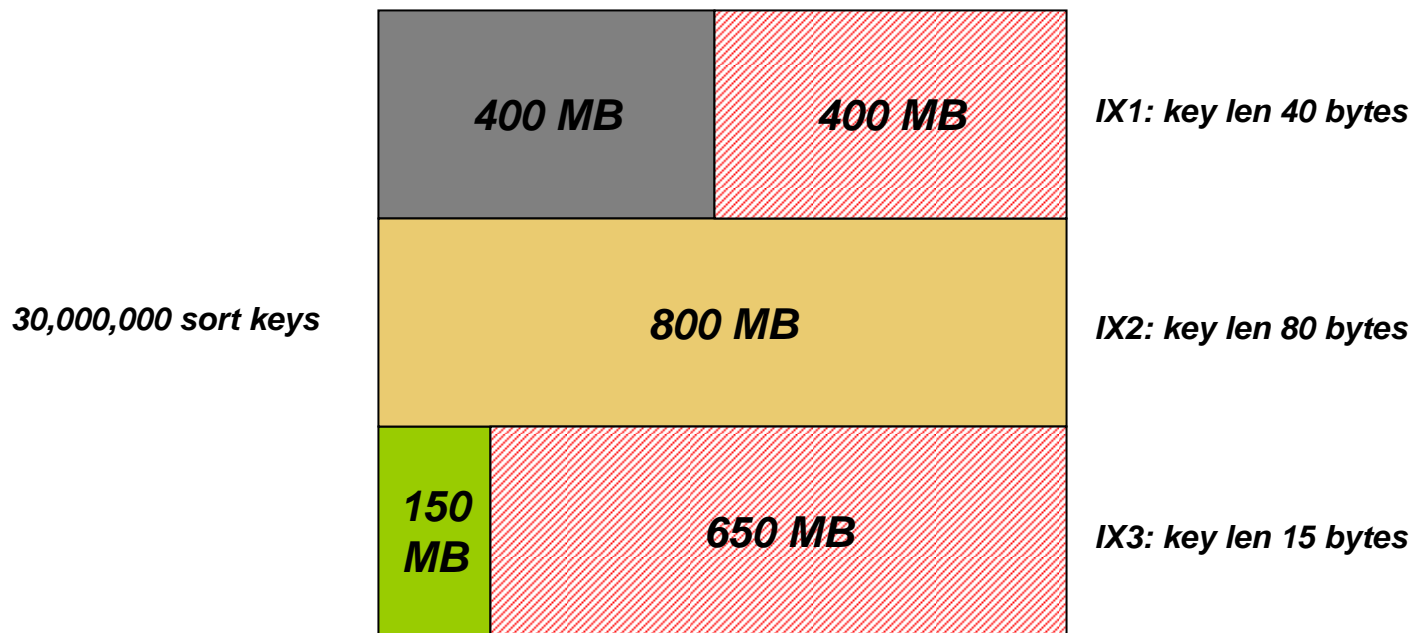


Required space: 1,600 MB (250 MB overhead)

# Sample: Combining Multiple Sort Tasks (2)



- Fully constrained system: 1 task for 3 indexes



Required space: 2,400 MB (1,050 MB overhead)

# Improved Sorting Estimates with Real-Time Statistics



- For some sorts the number of records to be sorted has to be estimated
  - Current logic estimated the number using RUNSTATS values
  - With UTSORTAL=YES we now lookup Real-Time Stats values first and only use the current logic as fall back if no RTS available
  - Message „DSNU3343I - REAL-TIME STATISTICS INFORMATION MISSING FOR ...” issued if RTS values not available
- Required RTS values are initialized by REORG TABLESPACE, LOAD REPLACE, and REBUILD INDEX, and of course automatically for all new objects
  - Looks like a catch-22, but we simply use the previous estimation logic, if info from RTS is not available
- If RTS not available for a specific object type, we use fall back logic:
  - Use index (partition) RTS value instead of table space (partition) RTS, or vice versa
  - Use sum of one index RTS counts for each table in table space for table space stats
  - Not for XML objects as there is no 1:1 relationship
- Don't replace DB2 objects outside DB2's control to maintain RTS accuracy (e.g. DSN1COPY or DFSMS copy) or let DB2 “know”:
  - Set TOTALROWS in SYSIBM.(SYS)TABLESPACESTATS or TOTALENTRIES in SYSIBM.(SYS)INDEXSPACESTATS to NULL to invalidate existing statistics or to the known number of rows/keys if replacing with significantly different data

# Additional Thoughts for SORTNUM Elimination



- Overflow storage group for sort pool with less performant volumes doesn't fit the new concept of trying to allocate sort work data sets as large as possible
- Keeping Real-Time Stats accurate is beneficial
- Average row length as determined by RUNSTATS is also important to have reasonable values due to slightly reduced safety buffer in allocated sort work data set size
- Real-Time Stats are not updated by RUNSTATS
  - Usually RUNSTATS is run SHRLEVEL CHANGE which only operates on a shadow copy with no log phase, so counts are never of the „quality“ needed by RTS

# SORTNUM elimination summary



- SORTNUM Elimination can help you to run all your sorting utilities more effectively
- Real-Time Statistics improve sort work space allocation accuracy
- Best Practices to improve your environment to suit the new functionality

# SORTNUM Elimination Related Maintenance



- PK45916 (UK33692 - PUT0802, RSU0806): SORTNUM elimination shipment for DB2 for z/OS V8
- PK41899 (UK33636 - PUT0802, RSU0806): SORTNUM elimination shipment for DB2 9 for z/OS
- PK64624 (UK36436/UK36437 - PUT0805, RSU0806): Different abends in LOAD with multiple INTO TABLE
- PK64915 (UK36331/UK36332 - PUT0805, RSU0809): Improve estimates for REBUILD INDEX/CHECK INDEX with segmented table spaces and missing RTS
- PK66597 (UK37865/UK37866 - PUT0807, RSU0808): LOAD ABEND0C4 RC00000011 when SYSTEMPL DD specified but not used
- **PK70001 (UK39566/UK39568 - PUT0809, RSU0812): ICE046A SORT CAPACITY EXCEEDED when REORG TABLESPACE is restarted in UNLOAD phase, improved fall back estimates for multi table table spaces**
- PK71733 (UK41940/UK41941 - PUT0812, RSU0903): DB2 over allocates sort work data sets when data sets are written in many fragments
- **PK75022 (UK42014 - PUT0812, RSU0903): REORG TABLESPACE ABEND0C4 when no index defined on partitioned table space with unload/reload parallelism**
- **PK75647 (UK43070/UK43071 - PUT0901, RSU0906): MSGIKJ56246I FILE nnn NOT ALLOCATED, FILE IN USE with DFSORT option IGNWKDD**
- **PK77774 (UK43598/UK43608 - PUT0902, RSU0906): Reduce secondary quantity size for DB2 allocated sort work data sets, REORG TABLESPACE underestimates average row length for highly compressed table spaces**
- PK80947 (UK45425/UK45426 - PUT0904, RSUxx): Utility failure when RTS value contains negative values or values exceeding physical database limits
- **PK81133 (UK44993/UK44994 - PUT0904, RSUxx): REBUILD INDEX fails with ICE046A SORT CAPACITY EXCEEDED when partitioned index is REBUILD PENDING**
- **PK82367 (UK45427/UK45428 - PUT0904, RSUxx): TOTALROWS/TOTALENTRIES turn negative when exceeding 2,147,483,647**
- **PK84412 (UK47681 - PUTxx, RSUxx): Increased elapsed time in REORG TABLESPACE when using unload/reload parallelism with many partitions, or CHECK INDEX/REBUILD INDEX on a single partitioned index**
- **PK87579 (open): CHECK INDEX and REBUILD INDEX don't honor SORTWKnn DD cards as an override**
- PK90293 (open): REORG TABLESPACE ABEND04E RC00E40045 during partition unload/reload parallelism

# General Sort Related Maintenance

- PK59399 (UK39757/UK39811 - PUT0809, RSU0812): Support DFSORT installation options in PARMLIB members in z/OS 1.10
- **PK61785 (UK35830 - PUT0805, RSU0809): Average row length too small during REORG TABLESPACE with partition parallelism after alter added columns or when building a new compression dictionary**
- PK66300 (UK37122 - PUT0806, RSU0809): REORG TABLESPACE with partition parallelism doesn't account for keys in mapping table
- **PK76697 (UK42860/UK42861 - PUT0901, RSU0906): LOAD REPLACE into single table of a multi table table space underestimates numbers of keys to sort**
- PK79136 (UK44703 - PUT0903, RSU0906): Reduce auxiliary storage consumption in parallel DFSORT invocations
- PK82720 (UK46525/UK46527 - PUT0905, RSUxx): REBUILD INDEX fails for indexes on DSNDB06.SYSDBASE
- **PK83210 (UK46444/UK46445 - PUT0905, RSUxx): REORG TABLESPACE fails with ICE046A SORT CAPACITY EXCEEDED with empty compressed partitions**
- PK87586 (open): LOAD fails in index sort when checking non indexed foreign keys
- DFSORT APAR PK63409 (UK35433 - PUT0804, RSU0809): ICE046A SORT CAPACITY EXCEEDED when estimate is slightly less than actual value
  - Fix for DFSORT in z/OS 1.10 is provided in PK66899 (UK38900 - PUT0808, RSU0812)

# CHECK INDEX Best Practices



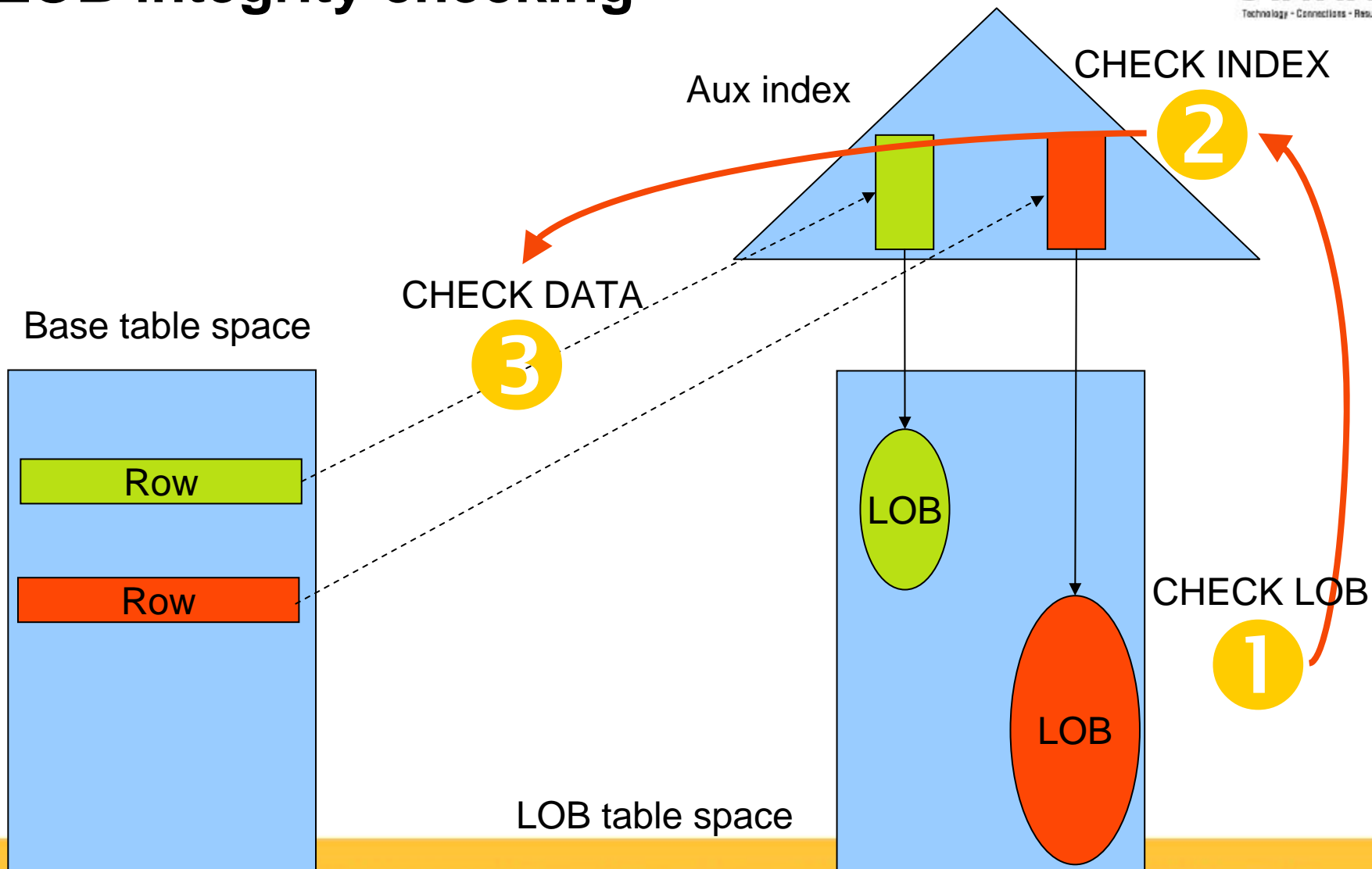
- CHECK INDEX
  - Indexes are checked in parallel
  - Use SHRLEVEL CHANGE
    - Uses dataset-level FlashCopy2 if available
    - Else, traditional media copy – still smaller r/o outage than SHR REF
  - PK41711 allows specification of storage class for shadow data sets
    - Useful in PPRC environments that shadow data sets can be placed on non-PPRC volumes
    - Defined in ZPARM UTIL\_TEMP\_STORCLAS

# CHECK DATA/LOB Best Practices



- CHECK DATA
  - If large volumes of delete data (e.g. after REORG DISCARD)
    - LOG NO to avoid log archive and log latch contention
    - Image COPY will be required
- CHECK DATA & CHECK LOB
  - DB2 9 adds SHRLEVEL CHANGE support:
    - Short term drain of writers to allow flashcopy to shadow
      - *Usual drain parameters supported*
    - CHKP/ACHKP/AUXW no longer set if errors detected
      - *Not reset either – use REPAIR*
      - *Look for messages and generated REPAIR statements*
  - CHECK DATA SHRLEVEL CHANGE cannot delete rows or mark LOBs invalid, it will write REPAIR statements to PUNCHDDN
    - *REPAIR LOCATE DELETE statements instead of RI discard*
    - *REPAIR LOCATE VERIFY/REPLACE statements to invalidate LOBs*
  - PK41711 for non-PPRC volumes to be used for shadow data sets

# LOB integrity checking



# DSN1COPY – what you need to know



- DSN1COPY is an essential part of the utilities portfolio
- DSN1COPY runs standalone and cannot ensure that data matches definition at target
- All target datasets must be preallocated for multi-piece tablespaces
- Areas to watch out for
  - BRF-RRF mismatch
    - Tolerated by SQL, but not REORG
    - Convert pagesets to ensure copy is RRF-RRF
    - No method exists today to convert RRF to BRF
  - Data definition changes, e.g. columns added
    - Use REPAIR VERSIONS at target site
    - For alterations prior to V8, REORG at source before DSN1COPY
  - Different tablespace types or different segsizes
    - Not policed, abends will occur
  - XML
    - Data-dependent information kept in catalog table XMLSTRINGS
    - Cannot DSN1COPY XML tablespace from one subsystem/group to another
    - DSN1COPY within a subsystem/group is fine
    - Solution is UNLOAD/LOAD/CROSSLOADER
    - Other problem:
      - DOCID is a sequence generated by DB2 – DSN1COPY to a new target where the DOCID is lower will result in -803 on insert because DB2 generates a value of n but n already exists in the table. ALTER of the sequence isn't allowed for DB2-generated sequences.
      - SELECT NEXT VALUE FOR <seq-for-docid> can be used to increment the number up to the max DOCID in the table

# RUNSTATS Best Practices



- RUNSTATS
  - SHRLEVEL CHANGE for availability
  - Collect only column stats on columns used in SQL predicates
    - Use the Statistics Advisor to detect which stats to collect
    - SAMPLE reduces CPU time when gathering column stats
  - KEYCARD provides valuable info for little processing cost

## Utilities On Demand

- Run utilities only when necessary and not on fixed schedules
- Information on the current status of all objects is contained in Real-Time Statistics (RTS) tables
- Stored Procedure DSNACCOR applies our suggested thresholds and formulas against a list of objects and recommends utility actions
- DB2 9 NFM adds Stored Procedure DSNACCOX (PK44133) with additional real-time statistics being used and improved recommendations

## IBM's UNLOAD Products

- Two UNLOAD utilities from IBM
  - DB2 UNLOAD Utility (in the IBM DB2 Utilities Suite)
  - DB2 High Performance Unload (HPU) Utility
  - (DSNTIAUL is only a sample!)
- HPU was delivered before the UNLOAD utility – had this not been the case, we would never have used the words “High Performance”
- In elapsed time, they are comparable (sometimes UNLOAD is faster, sometimes HPU is faster)
- In CPU time, HPU consumes approximately half the CPU in many situations (but not always)
- UNLOAD is geared towards user of DB2 Utilities (Utilities interface)
- HPU is geared towards application developers (SQL interface)

# LOB Handling in LOAD/UNLOAD w/FRVs



- Avoid 32K limit on LOAD/UNLOAD
- Need to allow user to limit LOAD at target with WHEN clause
- LOB column values will be stored as separate PDS member, PDS/E member, or HFS directory member.
- LOB column values from each row will have identical member names in each PDS, PDS/E, or HFS
- Data set name stored in output record
- Design fits well with application support for File Reference Variables in V9
- Apply PK75216 for significant performance enhancement for PDS FRVs in V9
  - HFS still performs better in elapsed time
- Note limit of 522,239 members in a PDS/E
- NULL LOBs are handled better than zero length LOBs
  - No FRV created on UNLOAD for null LOBs
- V8 – zero length LOBs will result in zero length LOBs put in LOB tablespace
  - Fixed in V9

# LOB Handling in LOAD/UNLOAD w/FRVs



- LOAD is changed to allow an input field value to contain the name of file containing a LOB column value. The LOB is loaded from that file.

```
//SYSREC DD *  
"000001" , "UN.DB1.TS1.RESUME(AI3WX3JT)" , "UN.DB1.TS1.PHOTO(AI3WX3JT)"  
"000002" , "UN.DB1.TS1.RESUME(AI3WX5BS)" , "UN.DB1.TS1.PHOTO(AI3WX5BS)"  
"000003" , "UN.DB1.TS1.RESUME(AI3WX5CC)" , "UN.DB1.TS1.PHOTO(AI3WX5CC)"  
"000004" , "UN.DB1.TS1.RESUME(AI3WX5CK)" , "UN.DB1.TS1.PHOTO(AI3WX5CK)"
```

```
LOAD DATA FORMAT DELIMITED  
INTO TABLE MY_EMP_PHOTO_RESUME  
  (EMPNO CHAR ,  
   RESUME VARCHAR CLOBF ,  
   PHOTO VARCHAR BLOBF)
```

new syntax

# LOB Handling in LOAD/UNLOAD w/FRVs



- UNLOAD is changed to store the value of a LOB column in a file and record the name of the file in the unloaded record of the base table.

```
TEMPLATE LOBFRV1 DSN `UN.&DB..&TS..RESUME` DSNTYPE(PDS) UNIT(SYSDA)
TEMPLATE LOBFRV2 DSN `UN.&DB..&TS..PHOTO` DSNTYPE(PDS) UNIT(SYSDA)
```

UNLOAD DATA

FROM TABLE DSN8910.EMP\_PHOTO\_RESUME

(EMPNO CHAR(6),

RESUME VARCHAR(255) CLOBF LOBFRV1,

PHOTO VARCHAR(255) BLOBF LOBFRV2) DELIMITED new syntax

Output:

"000001", "UN.DB1.TS1.RESUME(AI3WX3JT)", "UN.DB1.TS1.PHOTO(AI3WX3JT)"

"000002", "UN.DB1.TS1.RESUME(AI3WX5BS)", "UN.DB1.TS1.PHOTO(AI3WX5BS)"

"000003", "UN.DB1.TS1.RESUME(AI3WX5CC)", "UN.DB1.TS1.PHOTO(AI3WX5CC)"

"000004", "UN.DB1.TS1.RESUME(AI3WX5CK)", "UN.DB1.TS1.PHOTO(AI3WX5CK)"

...

# Provide logic for routine maintenance




- Leverage the ability to invoke utilities programmatically via stored procedures
  - DSNUTILS for EBCDIC parameters
  - DSNUTILU for UNICODE parameters

# Provide logic for routine maintenance

Example (using REXX):

```
/* REXX */
...
ADDRESS DSNREXX "CONNECT DB2P"
IF SQLCODE ^= 0 THEN CALL SQLCA
  Uid="";Restart=""; Utstmt= ,
  'REORG TABLESPACE' ,
  'ADHTSTDB.ADHTSTTS' ,
  'LOG NO KEEPDICTIONARY' ,
  'SORTDATA SORTKEYS SORTDEVT' ,
  'STATISTICS' ,
  'TABLE (T1) SAMPLE 25 COLUMN (C1, C2)' ,
  'TABLE (T2) SAMPLE 25 COLUMN (C5, C12)'
  Utility='REORG TABLESPACE'
  CopyDSN1='DSN.FIC.ADHTSTTS.VERSION(+1)'
  CopyDEVT1='SYSDA'
  CopySpace1=1
```



```
ADDRESS DSNREXX "EXECSQL" ,
"CALL DSNUTILS(:UID, :RESTART,           " ,
"      :UTSTMT,           " ,
"      :RETCODE,           " ,
"      :UTILITY,           " ,
"      :RECDSN ,:RECDEVT ,:RECSPACE ," ,
"      :DISCDSN ,:DISCDEVT ,:DISCSPACE ," ,
"      :PNCHDSN ,:PNCHDEVT ,:PNCHSPACE ," ,
"      :COPYDSN1,:COPYDEVT1,:COPYSPACE1," ,
"      :COPYDSN2,:COPYDEVT2,:COPYSPACE2," ,
"      :RCPYDSN1,:RCPYDEVT1,:RCPYSPACE1," ,
"      :RCPYDSN2,:RCPYDEVT2,:RCPYSPACE2," ,
"      :WORKDSN1,:WORKDEVT1,:WORKSPACE1," ,
"      :WORKDSN2,:WORKDEVT2,:WORKSPACE2," ,
"      :MAPDSN ,:MAPDEVT ,:MAPSPACE ," ,
"      :ERRDSN ,:ERRDEVT ,:ERRSPACE ," ,
"      :FILTRDSN,:FILTDEVT ,:FILTRSPACE)"
IF SQLCODE < 0 THEN CALL SQLCA
...
```

# Provide logic for routine maintenance

- Rich logic can be provided to:
  - Take an image copy before running REORG with NOSYSREC
  - Examine statistics (from RUNSTATS or the Real-time statistics) to determine when to run a utility (see DSNACCOR/DSNACCOX)
  - Examine a control table to determine windows when maintenance can or cannot be run
  - ...
- You have full control without needing individual threshold keywords on each utility
- But, maybe you don't want to write or maintain this type of logic yourself... that where products like the DB2 Automation Tool for z/OS come into play

# Summary

- Continuing commitment to, & investment in, utilities
- Support core DB2 function from day 1
- Ensure utilities are non-disruptive
  - Eliminate outages
  - Improve performance
  - Reduce CPU cost
- Provide function that adds real value
- Reduce complexity & improve automation
- Revisit your existing utility jobs to benefit from new options
- SORTNUM Elimination can help you to run all your sorting utilities more efficiently
- Use DB2 provided stored procedures to schedule utilities “On-Demand” instead of invoking them on fixed schedules