DB2 for z/OS Utilities Update: Best Practices

Haakon Roberts, IBM Silicon Valley Lab
haakon@us.ibm.com

Mar 2009
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

<table>
<thead>
<tr>
<th>Seq No</th>
<th>Low Value</th>
<th>High Value</th>
<th>Cardinality</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>5/12</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>9</td>
<td>4</td>
<td>4/12</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>15</td>
<td>3</td>
<td>3/12</td>
</tr>
</tbody>
</table>
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
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 X
- Better information for DPROPR/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
  - DELETE
  - AGE \(\text{integer}\)
  - DATE \(\text{integer}\) \(^{(*)}\)
  - RETAIN
  - LAST \((\text{integer})\)
  - LOGLIMIT
  - GDGLIMIT
  - GDGLIMIT \(\text{LAST} \ (\text{integer})\)
  - GDGLIMIT \(\text{LOGLIMIT}\)

  \(\rightarrow\) Queries SYSCOPY
  \(\leftarrow\) Queries BSDS
  \(\leftarrow\) Queries GDG
  \{\text{For mixed lists}\}

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

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

```sql
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=$100$MB, 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
  - KEEPDICTIORY (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
  - Avoid data conversion, use internal representation if possible
  - 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
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
  - Index parallelism (SORTKEYS is default and ignored in V8)
    - 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, no 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)
  - (DRAIN_WAIT+MAXRO)<(IRLMRWT -5 or 10 seconds)
  - Avoid application timeouts
  - But don’t set MAXRO too low
  - RETRY = utility lock timeout multiplier (6 by default)
  - RETRY_DELAY = DRAIN_WAIT*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
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 LOBs if performance degraded because of bad LOB chunking
- DB2 9 - use SHRLEVEL REFERENCE
  - 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 utility 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
- Recommendation:
  - View as single table and REORG as a whole

PBG

<table>
<thead>
<tr>
<th>Part 1</th>
<th>UTRW/UTRO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part 2</td>
<td>UTRW</td>
</tr>
<tr>
<td>Part 3</td>
<td>UTRW</td>
</tr>
</tbody>
</table>
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/DELETE allowed that affects unique index
    - No parallel jobs on different indexes of the same table space -> use single job with multiple indexes specified
Dynamically Allocated Sort Work Data Sets

- DB2/DFSORT determined DS sizes without DDs
- Single JCL (template) can be used for most utility jobs
- DB2 determines degree of parallelism according to available resources
- BUT:
  - Need to specify SORTNUM, but one size does NOT fit all
    - Different objects being processed by same job template
    - Different sorts within same utility, e.g. REORG with data and index sorts
  - DASD situation varies, SORTNUM 4 might work today, but tomorrow even SORTNUM 8 might fail
  - DB2’s estimates sometimes not good enough
DB2 Allocated Sort Work Data Sets

- PTFs shipped 02/2008 to enable DB2 to dynamically allocate sort work data sets in utilities:
  - DB2 for z/OS V8: PK45916 / UK33692
  - DB2 9 for z/OS: PK41899 / UK33636
  - Enable with UTSORTAL=YES
  - Used for all sorts in utilities: LOAD, REORG, CHECK INDEX, REBUILD INDEX, CHECK DATA, RUNSTATS
  - Message “DSNU3340I - UTILITY PERFORMS DYNAMIC ALLOCATION OF SORT DISK SPACE” indicates use
  - New behavior ignored if hard coded DD cards are found

- No more need to specify SORTNUM. Existing SORTNUM specification can be honored or ignored (IGNSORTN=YES)

- Data sets for largest sorts are allocated first

- Attempts to allocate data sets as large as possible (starting with 2 data sets per sort task, more data sets allocated if necessary)
DB2 Allocated Sort Work Data Sets

- Uses Real-Time statistics for size estimates
- Start using RTS on V8 if not already done (always active in DB2 9)
  - RTS can benefit you in many ways
- Required values in RTS are initialized by REORG TABLESPACE and REBUILD INDEX
- If replacing DB2 objects outside DB2’s control then notify DB2 that RTS information isn’t accurate:
  - Set TOTALROWS to NULL in SYSIBM.(SYS)TABLESPACESTATS or TOTALENTRIES to NULL in SYSIBM.(SYS)INDEXSPACESTATS to invalidate existing statistics if replacing with significantly different data
DB2 Allocated Sort Work Data Sets

- **Recommended maintenance:**
  - APAR PK64624: LOAD with multiple INTO TABLE
  - APAR PK64915: Improve estimates for REBUILD and CHECK INDEX with segmented table spaces with missing RTS
  - APAR PK66597: LOAD ABEND0C4 RC00000011 when SYSTEMPL DD specified but not used
  - APAR PK70001: ICE046A SORT CAPACITY EXCEEDED when REORG is restarted in UNLOAD phase, improved fall back estimates for multi table table spaces
  - DFSORT APAR PK63409: ICE046A SORT CAPACITY EXCEEDED when estimate is slightly below actual value
• **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

1. CHECK LOB
2. CHECK INDEX
3. CHECK DATA
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
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 (see next slide)
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 DSNACCCOR 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

- Requirement is to move LOBs from one z/OS system to another z/OS system
- Need to support millions of rows
- Typical LOB sizes are 25K, 200K, 1MB
- 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
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
```

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)’
CopyDEV1=‘SYSDA’
CopySpace1=1

ADDRESS DSNREXX "EXEC SQL",
"CALL DSNUTILS(:UID, :RESTART,",
" :UTSTMT,",
" :RETCODE,",
" :UTILITY,",
" :RECSN :RECDVS :RECSPACE ",",
" :DISCSN :DISCDVS :DISCSPACE ",",
" :PCHSN :PCHDEV :PCHSPACE ",",
" :COPYDSN1 :COPYDEV1 :COPYSPACE1 ",",
" :COPYDSN2 :COPYDEV2 :COPYSPACE2 ",",
" :RCPYDSN1 :RCPYDEV1 :RCPYSPACE1 ",",
" :RCPYDSN2 :RCPYDEV2 :RCPYSPACE2 ",",
" :WORKDSN1 :WORKDEV1 :WORKSPACE1 ",",
" :WORKDSN2 :WORKDEV2 :WORKSPACE2 ",",
" :MAPDSN :MAPDEV :MAPSPACE ",",
" :ERRDSN :ERRDEV :ERRSPACE ",",
" :FILTRDSN :FILTDEV :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 effectively
- Use DB2 provided stored procedures to schedule utilities “On-Demand” instead of invoking them on fixed schedules