DB2 for z/OS

Large Objects

Haakon Roberts

DB2 Development

haakon@us.ibm.com

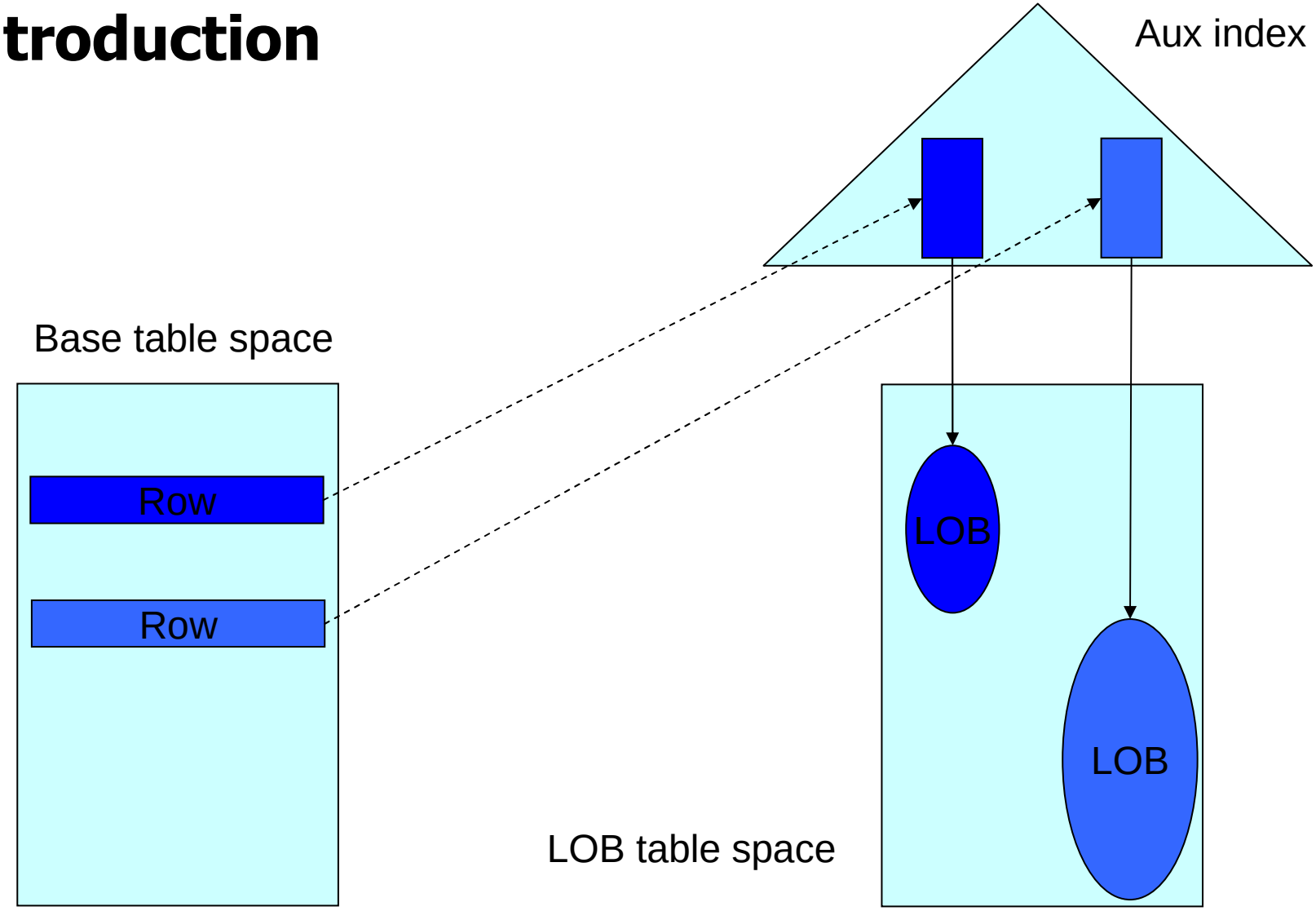
Important Disclaimer

- THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY.
- 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.
- IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE.
- IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION.
- NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, OR SHALL HAVE THE EFFECT OF:
  - CREATING ANY WARRANTY OR REPRESENTATION FROM IBM (OR ITS AFFILIATES OR ITS OR THEIR SUPPLIERS AND/OR LICENSORS); OR
  - ALTERING THE TERMS AND CONDITIONS OF THE APPLICABLE LICENSE AGREEMENT GOVERNING THE USE OF IBM SOFTWARE.

Agenda

- Introduction
- Locking enhancements
- Application Flow Optimization
- File Reference Variables
- LOAD/UNLOAD/Crossloader enhancements
- REORG enhancements
- CHECK LOB/DATA enhancements
- Other
- Future enhancements & reference

Introduction



The basics

- One LOB tablespace per LOB column per partition
  - 2 LOB columns & 4096 parts = 8192 LOB tablespaces
- One aux table and one aux index per LOB tablespace
- Access to LOB data is via base row and aux index
  - No table space scan, no access path selection or optimization
- LOB uniquely defined by ROWID and version
  - Update changes version, ROWID never changes
- Internally, no concept of update of LOB
  - Update is deallocation of old version, allocation of new
- Can be LOG YES or LOG NO
  - LOG NO does not mean no logging
- Separate from base tablespace
  - REORG, COPY, RECOVER, RUNSTATS
- Keep in sync with base

The basics

- Application processing - choices
  - Process the LOB as a whole
    - Not possible for large LOBs
  - Process sections of the LOB
  - LOB locators
    - 4-byte representation of LOB, section of LOB or concatenation of LOBs

Locking enhancements

- Why?
  - Availability
    - LOB locks always held until commit
    - LOB locks acquired even for UR readers
      - Lock escalation can occur on LOB tablespace
  - Performance
    - Significant performance overhead
    - LOB locks acquired for space search purposes

Locking enhancements

- INSERT
  - Pre-V9
    - Get X lock on LOB, hold until commit
  - V9
    - Get X lock on LOB, release after allocation complete
    - In data sharing, need to ensure changed pages are forced out before lock is released, therefore recommend GBPCACHE(CHANGED) for improved performance
- DELETE
  - Pre-V9
    - Get S lock on LOB, hold until commit
  - V9
    - No LOB lock acquired

Locking enhancements

- SELECT
  - Pre-V9
    - Get S lock on LOB, hold until commit
  - V9
    - For non-UR readers, no LOB lock acquired
    - For UR readers, get S LOB lock & release immediately
- UPDATE
  - Pre-V9
    - Update is delete followed by insert
  - V9
    - Same
      - Benefit from reduced locking for insert, delete & space search

Locking enhancements

- Space search for LOB allocation
  - Pre-V9
    - If a free page belongs to a deallocated LOB then
      - Check if deallocation occurred prior to oldest read claim
        - This test not particularly granular
      - If readlsn checking fails then try to get lock on old LOB
        - If successful, use the page, else try another
    - Can result in many lock requests for a single LOB allocation
  - V9
    - No LOB locks acquired for space search
    - Readlsn granularity improved to page level in LOB table space
      - >=192x more granular

Locking enhancements - summary

- Significant reduction in locking overhead for LOB processing
- Improved availability & performance
- Recommend GBPCACHE(CHANGED) for improved performance
- Requirements
  - NFM
  - "Locking protocol 3"
    - Automatic in non-data sharing
    - Clean group-wide shutdown in data sharing once NFM enabled
  - PK62027 – remove requirement for clean group-wide shutdown

Application flow optimization

- Why?
  - Need to reduce flow of unnecessary LOB data for small/medium sized LOB data requests across a network
  - Need to simplify manipulation of large LOBs
  - LOB data transfer currently optimized for large amounts of data
  - Locators may result in poor performance
    - More resource consumption at server
      - Particularly if locator not freed or application doesn't commit
    - More complex application coding
      - Up to 3 trips to DB2 to materialise LOB
  - For large LOBs, locators often used, but this incurs a separate network flow before data is retrieved
  - For small LOBs, more efficient to retrieve LOB data directly
  - Increased application virtual storage consumption if XML or locator not used
    - Maximum buffer size must be allocated if want to avoid truncation
    - For XML documents, maximum size not known, so educated guesswork
  - Improve handling of large XML objects
    - No locator support for XML

Application flow optimization

- Progressive streaming
  - Provide new LOB/XML data retrieval design
    - Effective for small/medium size objects
    - More efficient use of locators to retrieve large amounts of data
  - Introduce ability for server to dynamically determine most efficient method to return LOB/XML data
  - With dynamic data format enabled, locator is kept for lifespan of cursor, not transaction
  - JDBC, SQLJ, and CLI will let server determine whether to flow LOB values or locators based on size thresholds
    - 2 new JCC T4 datasource properties
      - progressiveStreaming
      - streamBufferSize

Application flow optimization

- FETCH CONTINUE
  - Retrieve LOB or XML data in multiple pieces without use of locators
  - Continue fetch of remaining data when truncation occurs
  - Must specify WITH CONTINUE on initial FETCH
  - Subsequent fetches use FETCH CURRENT CONTINUE
  - Application must manage buffers & reassemble data
    - Not required to fetch entire object before moving to next
  - SQLCA indicates whether data is truncated
  - No multi-row fetch support
  - Universal JDBC driver exploits this to implement progressive streaming
- Locators still recommended for random access to subset of object or if materialisation is to be avoided

Application flow optimization - summary

- Significant reduction in network traffic
- Simplified application design
- Improved performance
  - More efficient retrieval of small/medium LOBs avoiding use of locators
- Improved resource utilization for locators
- Reduced application virtual storage consumption
- Requirements
  - NFM
  - JDBC/CLI/SQLJ dependency also

File reference variables

- Why?
  - Difficult to load/unload large LOBs
  - Poor performance on load/unload
  - Significant application working storage requirement when manipulating large LOBs
  - Cross-platform compatibility

File reference variables

- FRVs defined in a host language
  - Contains file name and allows direct transfer of LOB data between DB2 and the file
- Language support
  - C, C++, Java
  - COBOL, PL/1
  - Assembler
  - REXX
  - No FRV implementation required in JCC type 2 or type 4 drivers
    - Both JCC Type 2 and Type 4 drivers have the capability to read a file to be used in its 'stream' (byte/character) methods

File reference variables

- Allow a large LOB or XML value to be inserted from a file or selected into a file rather than a host variable
- Application no longer needs to acquire storage to contain the LOB or XML value
- Bypass host language limitations on the maximum allowed size for LOB values located in working storage
- Support HFS or BSAM
  - HFS used if filename contains a "/", otherwise BSAM
- FRVs cannot be used as parameters for stored procedures or UDFs
- New SQL host variables
  - BLOB_FILE
  - CLOB_FILE
  - DBCLOB_FILE

File reference variables - summary

- Significantly improve DB2 capability for handling large LOBs
  - Improved performance
  - Improved usability
- Reduce application virtual storage requirements
- Improve family compatibility & application portability
- Requirements
  - V9

LOAD/UNLOAD/Crossloader

- Why?
  - LOAD utility limitation of 32Kb for input row length
- Limitation removal for Crossloader
  - Separate buffer used for LOB column values
  - DSNU1778I only issued if
    - (Sum of lengths of non-LOB columns) + (8 x Number of LOB columns) exceeds 32Kb or
    - Sum of lengths of LOB columns exceeds half of available memory above the line
  - PQ90263 in V7/V8
- LOAD/UNLOAD support for file reference variables
  - PK22910 in V7/V8

REORG

- Why?
  - Pre-V9, REORG of LOB table spaces has a number of drawbacks
  - No physical space reclamation
  - SHRLEVEL NONE only
    - No access to LOB data during REORG
  - Re-chunking of LOB data may be sub-optimal
    - Trade-off between space consumption & reorganization
  - LOG YES only
    - May result in excessive logging
  - Complex, susceptible to software defects

REORG

- Introduce SHRLEVEL REFERENCE option for REORG of LOB data
  - LOG NO only permitted option with SHRLEVEL REFERENCE
  - REORG will now load LOBs to shadow LOB pageset
    - Additional DASD space temporarily required
  - Available in both CM and NFM
  - Improved availability
  - Complete reorganization of LOB data
  - Full read access permitted to LOB data except during SWITCH phase
  - Inherit drain options from "standard" REORG
  - Inline imagecopy required to maintain recoverability
  - SHRLEVEL NONE still supported
    - Remains default, but will be deprecated in future
  - No restart capability
    - Shadow pageset discarded in event of failure

REORG - summary

- REORG of LOB data now much more viable
- Improved availability during REORG
- Improved LOB retrieval performance after REORG
- Physical space reclamation now possible
- Improved solution stability
- Requirements
  - V9 CM
  - Utility job JCL change

Other enhancements

- Allow logging of LOB data when max size > 1Gb
  - V9
  - Previously, LOG YES restricted to max size <=1Gb

Future enhancements & reference

- Improved performance
  - More efficient management of small LOBs
  - Improved streaming of large LOBs
- Further availability enhancements
- Further removal of usability restrictions

- Reference:
  - SG24–7270-00 LOBs with DB2 for z/OS: Stronger and Faster
  - www.ibm.com/redbooks