Performance and Availability
DB2 11 for z/OS
Disclaimer PLEASE READ THE FOLLOWING NOTICE

- The information contained in this presentation is based on techniques, algorithms, and documentation published by the several authors and companies, and in addition is the result of research. It is therefore subject to change at any time without notice or warning.
- The information contained in this presentation has not been submitted to any formal tests or review and is distributed on an “As is” basis without any warranty, either expressed or implied.
- The use of this information or the implementation of any of these techniques is a client responsibility and depends on the client’s ability to evaluate and integrate them into the client’s operational environment.
- While each item may have been reviewed for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere.
- Clients attempting to adapt these techniques to their own environments do so at their own risks.
- Slides, handouts, and additional materials distributed as part of this presentation or seminar should be reviewed in their entirety.
Abstract

With every new release of DB2 we look to see what features will allow us to improve the performance of our existing applications and the availability of our data. We also begin to plan to utilize new feature in our development efforts. This presentation takes a look at the features in DB2 11 that will both improve our performance and provide us with maximum data availability.

- Brief discussion of overall performance objectives of DB2 11
- Discuss details on database performance features and usage
- Discuss SQL, application and optimization performance enhancements
- Discuss considerations for implementing new features and enhancements
- Review availability enhancements and usage
Database and Subsystem Performance Enhancements

- Suppress null indexes
- Pseudo-deleted index entry cleanup
- Help for indirect references
- Large number of partitions access improvement
- No log declared temporary tables
- Incremental bind avoidance for DGTTs
- Parallelism support for DPSIs
- Partition elimination on join predicates
- Selective decompression
- RTS immediate externalization
- DSNACCOX improvements
- Access command parallelism
- Statistics feedback
- Work file usage improvements

- Reclassification of buffer pool pages
- 2G framesize support
- MRU queuing support for utilities
- New accounting for buffer pools
- New RBA/LRSN format
- Subsystem parameters for performance
Database Availability and Data Sharing

- RELEASE(DEALLOCATE) thread interruption
- Cancel DDF threads
- Drop column
- Online alter limit key
- Deferred definition enhancement
- PIT Recovery for deferred schema changes

- Restart light with CASTOUT option
- Conditional propagation of child update locks
- Improved performance in handling lock waiters
- Increase in maximum number of CF lock table entries
- Throttle batched unlock requests
- Improved IRLM resource hash table algorithm
- Group buffer pool write-around protocol
- Improved castout processing
- Improved DELETE_NAME performance
- Automatic LPL recovery at end of restart
- Log record sequence number spin avoidance
Utilities Performance and Availability

- Improved performance of partition-level REORG w/NPSI
- SWITCH phase impact reduction
- Physically delete empty partition-by-growth partitions
- Automated REORG mapping table management
- REORG change of defaults to match preferred practices
- REORG without SORTing data
- Partition-level inline image copy
- Improved REORG LISTDEF processing
- REBALANCE enhancements
- REORG of LOB enhancements
- Improved REORG serviceability
- RUNSTATS access path reset
- LOAD SHRLEVEL NONE/CHANGE with PARALLEL
- Addition of crossloader support for XML
- More offload to zIIP with NPSIs
- SYSLGRNX recording for catalog and directory
- RESTORE SYSTEM VCAT name translation
- Removal of incompatibility of REORG and COPY
- Removal of point-in-time recovery restrictions
- REPAIR Catalog
SQL and Optimization

- Predicate selectivity overrides
- APREUSE – WARN option
- Explain table enhancements
- Virtual index enhancements
- RID overflow processing
- Stage 2 predicate pushdown
- Correlated subquery conversion
- In memory sorts
- Duplicate removal during sort avoidance
- GROUP BY/DISTINCT early out for joins
- Early out processing
- Sparse index/hash joins
- Correlated subquery cache
- Invariant expression optimization
- Optimizer cost adjustments
- GROUPING SETS, ROLLUP and CUBE
Application, XML and Distributed Enhancements

- Global variables
- ARRAY data type performance
- Autonomous SQL procedures
- Transparent archiving
- Cancel thread
- Cancel SQL
- Column processing
- Data type processing
- Rollback to savepoint

- Package based continuous block fetch
- Implicit commit for stored procedures
- DDF synchronous receive
- Multi-threaded Java stored procedures
- JDBC/ODBC stored procedure optimizations
- Cancel DDF thread improvement

- Insert DOCID randomization
- Revalidation avoidance on LOAD
- XMLTABLE
  - Remove unreferenced column definitions
  - Merge common column path expressions
  - Storage reuse for output XML columns
  - Date/Time predicate pushdown
  - Optimize index key range
  - Pushdown of column casting
- Binary validation
Performance and Availability ‘Opportunities’

- When it comes to achieving the best performance and availability possible in DB2 we have to consider the following
  - Expectations
    - What new features look promising?
    - What problem are you looking to resolve with a new feature?
    - Is it a better option than what you are doing today?
  - Reality
    - What effort is required to take advantage of new features?
    - Will the usage achieve my goals?
    - What features will be automatic and did their implementation hurt or harm my current performance?
  - Usage
    - To use some new features there may be large changes needed
      - Rebinds, code changes, database changes
    - Plan for efforts needed
    - Evaluate effectiveness
Rebinds and Usage for Performance Improvements – DB2 11

- Rebinds and Possible Changes Needed
  - In-memory usage/caching
  - Select list non column expressions executed once
  - RID overflow to work file usage for set functions
  - DPSI page range screening join/correlation predicates and parallelism
  - Statistics collection and feedback
  - Filter factor hints
  - Query rewrite for predicate indexability
  - Stage 2 predicates pushdown(indexable)
  - Merging of predicates with views/TEs
  - Duplicate removal
  - Reduction of indirect references
  - No log DGTTs

- No Rebinds Needed
  - Sort performance
  - Automatic index pseudo delete cleanup
  - Data decompression performance
  - DPSI performance for merge
  - Improvements with large number of partitions
  - XML performance
  - RELEASE(DEALLOCATE) optimization
  - DGTT avoidance of incremental binds
  - ROLLBACK TO SAVEPOINT performance
  - Suppress-null indexes
  - xPROCs above 2GB bar
  - ACCESS DATABASE command performance
  - Data sharing/GBP improvements
Database Performance
Suppress Null Indexes

• Issue
  • DB2 must index every data row when creating an index
    • Affects performance and size of index
    • It is useful to exclude one or more values from being indexed
      – Values never used in a query (i.e. NULL, blank, 0)

• DB2 11 (NFM)
  • Improves insert performance of NULL entries
    • Option of excluding NULL rows from indexes
    • Index entries not created when all values for indexed columns are NULL
  • EXCLUDE NULL KEYS on CREATE INDEX
  • Reduced index size
  • Improves insert/update/delete performance
  • CREATE INDEX performance should also improve
  • RUNSTATS utility collect statistics only on non-NULL value
  • Table statistics derived from index are adjusted by number of excluded NULL values

  • Statistics will be same whether derived from a table scan, an EXCLUDE NULL KEYS index, or INCLUDE NULL KEYS index
**Pseudo-Deleted Index Entries - Issue**

- **Issue:**
  - Index entries are not physically deleted when rows are deleted
    - Unless delete operation has exclusive control of index page set
  - Marked as pseudo-deleted
  - Referred to as pseudo-deleted index entries
  - Pseudo-empty index pages contain only pseudo-deleted index entries
  - DB2 attempts to clean up pseudo-empty index pages during DELETE
    - If some of pseudo-deleted entries in page are not committed during DELETE
      - Cleanup cannot be performed
      - Some pseudo-empty pages are likely not cleaned up

<table>
<thead>
<tr>
<th>Account</th>
<th>Acc_Num</th>
<th>Balance</th>
<th>Type</th>
<th>User_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x 1234</td>
<td>400.00</td>
<td>Check</td>
<td>1111</td>
</tr>
<tr>
<td></td>
<td>3456</td>
<td>600.00</td>
<td>CD</td>
<td>5662</td>
</tr>
<tr>
<td></td>
<td>x 5672</td>
<td>800.00</td>
<td>Check</td>
<td>4579</td>
</tr>
<tr>
<td></td>
<td>4567</td>
<td>200.00</td>
<td>Saving</td>
<td>2222</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Index**

- IK1: x RID1
- IK2: RID2
- IK3: x RID3
- IK4: RID4
- IK5: RID5
- IK6: RID6

© YL&A 1999-2015
Pseudo-Deleted Entries – Issue (cont..)

- Index entries are marked pseudo-deleted
  - To handle a combination of other processes using index access
  - And potential roll back of deleted rows
- Subsequent searches continue to access these pseudo-deleted entries
  - Gradually degrade performance as more rows are deleted
- Pseudo-deleted index entries can also result in time-outs and deadlocks
  - For applications inserting data into tables with unique indexes
- Large amount of update activity can experience inconsistent performance
  - Need to REORG your tables and indexes
  - Monitor PSEUDO_DELETE_ENTRIES in SYSINDEXPART
    - If >10% of total index pages – need index reorg
  - Monitor NPAGES in INDEXSPACESTATS
    - Number of pages with only pseudo-deleted entries
  - Monitor REORGPSEUDODELETE in INDEXSPACESTATS
    - Number of pseudo delete entries since last REORG
- Average transaction response time increases until a REORG INDEX is done
  - Increased getpages, lock requests, CPU
Automatic Pseudo Deleted Index Entry Cleanup

- In DB2 11(CM)
  - In addition to any cleanup previously performed
    - DB2 automatically deletes pseudo-empty index pages and pseudo deleted index entries
    - Independent of SQL DELETE
  - Cleanup is performed only on indexes that have been opened for INSERT/DELETE/UPDATE by other DB2 processes
  - Pseudo deleted entries can be detected by SQL queries or INSERT/DELETE/UPDATE processes
  - Can be a large number of pseudo deleted entries in an index
    - If index is not already opened for INSERT/DELETE/UPDATE
      - Cleanup does not happen
  - DSNZPARM - control number of concurrent cleanup tasks
    - On by default (1)
    - Can also be controlled via catalog table

<table>
<thead>
<tr>
<th>INDEX_CLEANUP_THREADS</th>
<th>SYSINDEXCLEANUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 0</td>
<td></td>
</tr>
</tbody>
</table>

Runs under zIIP eligible SRBs
Free Space Issue for Updates

- On partition or table space level
  - Free space is defined
  - CREATE/ALTER TABLESPACE
- Reserved during LOAD or REORG processing
- Used by insert but not update
- Insert can consume all free space if available
  - Page is marked full when reaches designated percentage
  - Nothing left for UPDATEs
  - No easy way to managed free space for updates
  - Causing indirect reference
    - Driving more frequent REORGs
Indirect Reference Indicators

- RTS – SYSTABLESPACESTATS and SYSINDEXSPACESTATS
  - REORGNEARINDREF
    - Number of overflow records created and relocated near pointer record
  - REORGFARINDREF
    - Number of overflow records created and relocated far from pointer record

_Since last run of REORG or LOAD REPLACE, or object was created_

- Catalog – SYSTABLEPART
  - NEARINDREF
    - Number of rows relocated far from their original page
  - FARINDREF
    - Number of rows relocated far from their original page

<table>
<thead>
<tr>
<th>Indirect Refs</th>
<th>Near Increase</th>
<th>Far Increase</th>
<th>Row #</th>
<th>Count Increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>6,606,093</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>38,622</td>
<td>38,622</td>
<td>82,048</td>
<td>6,967,418</td>
<td>361,325</td>
</tr>
<tr>
<td>38,638</td>
<td>16</td>
<td>103,291</td>
<td>6,977,109</td>
<td>9,691</td>
</tr>
<tr>
<td>38,645</td>
<td>7</td>
<td>58,575</td>
<td>6,988,602</td>
<td>11,493</td>
</tr>
</tbody>
</table>
Help for Reducing Indirect References

- **Issue:**
  - Updates to variable length and/or compressed rows can increase length
  - If not enough space on data page
    - Row is relocated to another data page
    - Replaces original row with a pointer record
    - Index entries continue to refer to original row (RID)
  - Indirect references can cause additional I/O
    - To read extra data page into buffer pool
  - REORG TABLESPACE
    - Removes indirect references

- **DB2 11**
  - PCTFREE FOR UPDATE attribute on table space/partition
    - Reserves free space for updates
  - DSNZPARM – PCTFREE_UPD
    - Default = 0
    - -1 = DB2 uses RTS to determine setting
  - SYSTABLEPART – PCTFREE_UPD(defined)
    - PCTFREE_UPD_CALC(Calc’d by DB2 or utilities)
Large Number of Partitions - CPU Improvements

- **Issue**
  - There is CPU overhead when using RELEASE(COMMIT) when a large (>200) number of partitions exist on the table

- **DB2 11(CM)**
  - When using RELEASE(COMMIT) against a table with a large number of partitions (>200)
    - May experience some performance improvements
  - Performance is not sensitive to the number of partitions defined
    - Only sensitive to the number of partitions referenced under an individual COMMIT scope
    - The larger number of partitions, the larger the performance improvement observed
  - Improvement is found for applications issuing a single SELECT statement that touches only one partition out of a large number of partitions (within a commit scope)
  - Does not effect programs using RELEASE(DEALLOCATE)
DPSI Parallelism – Part-Level Join

- Prior to DB2 11
  - Joins with partitioning results in a large amount of random I/O
    - Each partition is probed on inner table of join
- DB2 11 (CM after rebinding)
  - Parallelism for improved join performance
    - When partitioned table space is inner table of a join
    - And partitioning on columns not included as join predicates
  - Part-level join
    - Each parallel child task processes only one partition, and composite (outer) table is shared or replicated to each child task
    - Each child task acts like it is a two-table join(involving one partition) rather than a join to multiple partitions
- Concurrent DPSI I/O access
  - Benefit very short running SQL which use DPSI merge access
  - When multiple DPSI parts are probed one after another
  - Concurrent I/O requests from qualifying DPSI parts
    - Reduces elapsed time
  - PARAMDEG_DPSI - Max degree of parallelism with a DPSI
DPSI – Partition Elimination on Join Predicates

• Prior to DB2 11
  • Partition elimination only works if …
    • There is a local predicate (literal value, host variable, parameter marker, special register), on the leading partition limit key(s)
    • Or, a local predicate can be transitivity closed against the leading partition limit key(s)
  • Could not be resolved with a join predicate

• DB2 11
  • Partition elimination on join predicates
    • Only qualified partitions are accessed
      – For a table partitioned on join columns
        • Each inner table probe only accesses qualified partitions
      – For table partitioned on non-join columns, and index is a DPSI
        • Each DPSI partition is processed sequentially

SELECT *
FROM CUSTOMER C, CUST_ORDER CO
WHERE C.CUSTID = CO.CUST_ID
AND C.CUST_DATE = CO.ORDER_DATE

Can be used for partition elimination
No Log Declared Global Temporary Table

• Prior to DB2 11
  • A declared global temporary table (DGTT)
    • Often used to store intermediate SQL results data
    • Overhead for logging of any insert/update/delete activity to DGTTs
• DB2 11(NFM)
  • Allows the option to avoid logging
    • During insert, update, and delete activity to DGTTs
  • Can improve the performance and usability of DGTTs
  • Maybe better to use DGTTs instead of a created global temporary table (CGTT)s
    • CGTT do not log, but do not support indexes
  • Compatible with DB2 family

DECLARED GLOBAL TEMPORARY TABLE tab1…..
NOT LOGGED
ON ROLLBACK DELETE ROWS
ON ROLLBACK PRESERVE ROWS
Incremental Bind and Re-Prepare Avoidance for DGTTs

Prior to DB2 11
- Incremental binds or re-prepares were needed for some SQL using declared global temporary tables

DB2 11
- Improves performance of using declared global temporary tables (DGTTs)
  - With certain SQL
  - By removing need for incremental bind (for static SQL)
    - or re-PREPARE (for dynamic SQL) after COMMIT
- SQL statement is kept in an executable ready state past COMMIT
  - Subsequent statement execution doesn’t need incremental bind or another PREPARE
- Occurs when application is bound with RELEASE(DEALLOCATE)
- For dynamic SQL
  - Unnecessary repeated PREPARE is removed
- Improvement is most noticeable for very short running SQL using a DGTT and frequent COMMITs
Selective De-Compression

Issue:
- DB2 uses row level hardware compression
  - Each row is decompressed before passed from buffer pool to application
  - Regardless of number of columns referenced by SQL statement
- When a query or utility accesses many rows, CPU can be high

DB2 11 (CM-Rebind)
- Optimizations introduced to reduce this CPU overhead
- Partial decompression, requires REBIND
  - Only applies to table space in reordered row format (RRF)
  - Builds a copy of expansion dictionary on first access to an object
    - Only decompresses portion of row needed starting from first byte
    - Columns for predicate evaluation are decompressed first
      - Because they are filtering rows
      - Additional columns in SELECT are decompressed if row qualifies
  - CPU reduction for table space scan of very large tables
    - When only a small percentage of columns are referenced
      - With predicate filtering, only qualifying table rows are decompressed
Work File Enhancement – Declared Temporary Tables

- Prior to DB2 11
  - Issues with separation of work files
  - DGTTs need to use separate space and can run out
- DB2 11
  - Warning messages can be issued when work file database space usage approaches a critical level
  - Ability to monitor space used by DGTTs and other work separately
  - WFSTGUSE_AGENT_THRESHOLD
    - Define the agent-level space-usage alert threshold
    - Percentage of available space in work file database on a subsystem that can be consumed by a single agent
      - Before DB2 issues a warning message
      - 0 (default) – 100
        - 0 = not used

WFSTGUSE_AGENT_THRESHOLD
Declared Temporary Tables – Space Thresholds (cont..)

- Issue a warning message when total amount of in-use work file storage (including DGTT) has reached a storage shortage threshold
  - (e.g. 10% of total configured storage)
- Once the warning message has been issued
  - DB2 will wait for 30 system check points before issuing another same warning message
- **WFSTGUSE_SYSTEM_THRESHOLD**
  - System level space usage alert threshold
  - Online-updateable
  - 0 to 100 (default 90)

**WFSTGUSE_SYSTEM_THRESHOLD**
Externalizing RTS Statistics

- Issue
  - RTS externalized in SYSTABLESPACESTATS/SYSINDEXSPACESTATS
    - Every 30 minutes (default)
      - Statistics are 15 minutes old (average) when tables are accessed
    - Changing objects may not reflect accurate information for tools (or users) making recommendations based on RTS (i.e. DSNACCOX)
  - DB2 11
    - Provides a way to externalize RTS
    - DB2 ACCESS command new option – MODE(STATS)
      - Users can trigger externalization of in-memory RTS blocks
    - MODE(STATS)
      - Externalizes in-memory statistics to RTS tables
        - In data sharing, externalized for all members
        - Does not physically open or change states of page sets

-ACCESS DB(DB2DB) SP(DB2TS) MODE(STATS)

-ACCESS DB(*) SP(*) MODE(STATS)
Externalizing RTS Command Options

Partition level statistics

ACCESS DB(db-name) SP (sp-name) MODE (STATS) PART (part-num)

Table space level statistics

ACCESS DB(db-name) SP (sp-name) MODE(STATS)

Database level statistics

ACCESS DB(db-name) SP (*) MODE (STATS)

Subsystem level statistics

ACCESS DB(*) SP (*) MODE (STATS)
ACCESS DATABASE Command Parallelism

- Prior to 11
  - Use of ACCESS DATABASE command to pre-open datasets was done
  - Moves overhead of physical open from an SQL thread to command thread
    - Transaction performance for first SQL thread to reference a given page set or partition is improved
  - Command was executed in serialized fashion
    - One task opening all data sets
      - Could be a long running processing
- DB2 11(CM)
  - Runs under a separate service task
  - Does not cause queuing of other database commands
  - Runs in parallel
    - Up to 20 threads
  - Elapsed time is significantly improved

-ACCESS DATABASE(DSN11*) SPACENAM(DSN11*) MODE(OPEN)
Bufferpool AUTOSIZE – MAX/MIN

- Prior to DB2 11
  - AUTOSIZE option introduced in DB2 10 to allow DB2 to use WLM to automatically increase buffer pool size as appropriate

- DB2 11
  - Ability to control the growth of buffer pool using AUTOSIZE
    - Minimum/maximum number of buffers to allocate (VPSIZE)

- AUTOSIZE YES – VPSIZEMIN/VPSIZEMAX
  - VPSIZE must be between VPSIZEMIN and VPSIZEMAX
  - VPSIZEMIN(*) or VPSIZEMAX(*)
    - Default 75% and 125% of current size respectively

```sql
> ALTER BUFFERPOOL (bpname) VPSIZE(integer)
  
  VPSIZEMIN(integer) VPSIZEMAX(integer)
  
  FRAMESIZE(4K) VPSEQT(integer) VPPSEQT(integer)
  
  PGFIX(NO) AUTOSIZE(NO)

<<
```
**Bufferpool AUTOSIZE – MAX/MIN**

- **DISPLAY BUFFERPOOL**
  - Shows maximum/minimum for size if AUTOSIZE used
  - 0 shown if not used

---

DSNB401I - BUFFERPOOL NAME BP0, BUFFERPOOL ID 0, USE COUNT 10
DSNB402I - BUFFERPOOL SIZE = 3000 BUFFERS AUTOSIZE = YES

VPSIZE MINIMUM = 2250     VPSIZE MAXIMUM = 3125

ALLOCATED = 3000          TO BE DELETED = 0
IN-USE/UPDATED = 200
BUFFERS ACTIVE = 3000
Bufferpool Getpage Classification

- Prior to DB2 11
  - The following were classified as random getpages
    - Dynamic/list prefetch and sequential format writes for utilities
- DB2 11
  - New classification of buffers as random versus sequential
    - Better aligned with prefetch and sequential format write operations
  - Now classified as sequential
    - Dynamic/list prefetch
    - Sequential format writes for utilities
  - MRU queuing available for utilities
    - Keeps used pages around, improving hit ratio
- To determine buffer hit ratio or page residency time
  - Need to know random buffer hit ratio and residency time for random pages
    - Random page residency time can be calculated from random hit ratio

Random residency time = max of these two formulas

\[
\text{Random residency time} = \max \left( \frac{\text{VPSIZE}}{\text{total pages read/second(including pages prefetched)}}, \frac{\text{VPSIZE} \times (1 - \text{VPSEQT}/100)}{\text{random synch I/O/sec}} \right)
\]

If this is larger - sequential getpages and prefetch are affecting residency time
Page Classifications – Display Bufferpool Detail

DSNB411I - RANDOM GETPAGE = 230
SEQ. GETPAGE = 610
DMTH HIT = 0
SYNC READ I/O (R) = 180
SYNC READ I/O (S) = 20
PAGE-INS REQ = 40

- SEQUENTIAL = 200
- RECLASSIFY = 0
- VPSEQT HIT = 0

DSNB412I - SEQUENTIAL PREFETCH -
REQUESTS = 0
PAGES READ = 0
PREFETCH I/O = 0

- SEQUENTIAL
  - # of buffers on SLRU chain
- RECLASSIFY
  - # times a random getpage touches a sequential buffer
    - May reclassify a sequential buffer as a random buffer
- VPSEQT HITS
  - # of times length of sequential queue reached VPSEQT

DB2 10 - PM70981
Increased Number of Open Datasets

- **Issue**
  - With the large increase in data sets needed there is a growing need for a larger number of open datasets
  - When DS MAX is reached (within 3%) datasets get closed
    - If reopened, can cause performance problems in applications

- **DB2 11 (CM)**
  - Increases the maximum number from 100,000 to 200,000
  - Practical limit might be significantly less on the DB2 subsystem
    - Depends on availability of virtual storage below 2 GB bar
    - Can be applied in DB2 10 via APAR PM88166
Overriding Predicate Selectivity

• Issue
  • Optimizer attempts to obtain lowest cost access path
  • Filtering by predicate is often hard to determine
    • Too many unknowns: skew, bad statistics, host variables, etc..
    • Results in poor access path selection

• DB2 11
  • Can override selectivity of predicates for matching statements
  • Predicate selectivity overrides
    • Selectivity values to be used instead of default filter factors during access path selection
      – Apply to all matching statements in specified context
  • Good for predicates whose selectivities are difficult or impossible to estimate
  • Improved selectivity information can be used to choose an optimal access path
    • Other methods (OPTHINT) enforce a particular access path
      – Remove DB2 query optimization from the process
  • Overrides simply provide additional information to optimizer

• Selectivity Profile
  • Used for filter factor stats collection for predicates
  • Populated by BIND QUERY
**DSN_PREDICATE_SELECTIVITY Table**

- **PREDNO**
  - Specific predicate in query
- **INSTANCE**
  - Selectivity instance
  - Groups related selectivities
- **WEIGHT**
  - % of executions that have specified selectivity
- **ASSUMPTION**
  - How selectivity was estimated or used
  - Normal – estimated using normal selectivity assumptions
  - Override – based on override

<table>
<thead>
<tr>
<th>INSTANCE</th>
<th>WEIGHT</th>
<th>PREDNO</th>
<th>SELECTIVITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.45</td>
<td>1</td>
<td>0.26</td>
</tr>
<tr>
<td>1</td>
<td>0.45</td>
<td>2</td>
<td>0.10</td>
</tr>
<tr>
<td>1</td>
<td>0.45</td>
<td>5</td>
<td>0.10</td>
</tr>
<tr>
<td>2</td>
<td>0.25</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>2</td>
<td>0.25</td>
<td>2</td>
<td>0.25</td>
</tr>
<tr>
<td>2</td>
<td>0.25</td>
<td>5</td>
<td>0.01</td>
</tr>
</tbody>
</table>

© YL&A 1999-2015
CREATE TABLE `userid`.`DSN_USERQUERY_TABLE`  
(QUERYNO INTEGER, SCHEMA VARCHAR(128), HINT_SCOPE SMALLINT, QUERY_TEXT CLOB(2M), QUERY_ROWID ROWID, QUERYID BIGINT, USERFILTER CHAR(8), OTHER_OPTIONS CHAR(128), COLLECTION VARCHAR(128), PACKAGE VARCHAR(128), VERSION VARCHAR(128), REOPT CHAR(1), STARJOIN CHAR(1), MAX_PAR_DEGREE INTEGER, DEF_CURR_DEGREE CHAR(3), SJTABLES INTEGER, OTHER_PARMS VARCHAR(128), SELECTVTY_OVERRIDE CHAR(1), ACCESSPATH_HINT CHAR(1), OPTION_OVERRIDE CHAR(1), ) IN `database-name`.`table-space-name`  
CCSID UNICODE;

**Selectivity override information**
APREUSE - WARN

Prior to DB2 11
- Access Path Reuse option on BIND/REBIND
- Avoid access path changes
  - At migration, fixes, application changes
- Applies to all statements in package
- Loads old access path and feeds as hint to optimizer
- Compares old/new access paths (implicitly turns on APCOMPARE)
- Determine REBIND success vs failure
  - APREUSE(ERROR) – works at package level
  - Some access paths may not be able to be reused
    - Failed access paths shown in PLAN_TABLE

DB2 11
- Introduces - APREUSE(WARN)
  - If reuse fails, optimizer generates new access path
  - Entire package does not fail if one SQL statement fails
    - Operates at statement level
  - Valid plan will be in PLAN_TABLE
Statistics Collection/Feedback

- Prior to DB2 11
  - What statistics to collect for best possible access path selection?
  - For each query it’s hard to know which statistics matter
    - For applications must know each SQL statement
    - Dynamic SQL it even more difficult
      - Dynamic statement cache can help, but is limited
  - Lack of proper statistics leads to inefficient access path
  - Optimizer uses statistics in catalog (not RTS)
    - No feedback provided regarding value of existing statistics
- DB2 11
  - Externalizes statistics recommendations for missing or conflicting statistics during optimization
  - Utilities(RUNSTATS) uses information as input to collect missing statistics
  - Feedback occurs at BIND, REBIND, and PREPARE
    - Externalized to SYSSTATFEEDBACK
  - During Explain
    - Externalized to DSN_STAT_FEEDBACK

SYSSTATFEEDBACK

DSN_STAT_FEEDBACK
Statistics Collection/Feedback

- Interpreting the statistics recommendations
  - Can use Optim Query Workload Tuner to generate statistics
  - Can manually create RUNSTATS jobs based on information in SYSSTATFEEDBACK
- Statistics recommendations can be at table, index or column level
- SYSSTATFEEDBACK table includes columns for identifiers
  - Additional columns in table contain information used to determine what statistics to collect
  - Use information in these columns to make decisions about what statistics to collect

- TYPE
  - Specifies the statistics to collect
- REASON
  - Identifies why the type of statistics were recommended
Statistics TYPE for Collection – Recommended Runstats

- **T** – Table

  ```sql
  RUNSTATS TABLESPACE ...TABLE(name)
  ```

- **I** – Index

  ```sql
  RUNSTATS INDEX
  ```

- **C** - Cardinality (single or multiple column – shown in NUMCOLUMNS)

  ```sql
  RUNSTATS TABLESPACE ...TABLE(name) COLUMN(name)
  ```

  ```sql
  RUNSTATS TABLESPACE ...TABLE(name) COLGROUP(name1,name2 ...)
  ```

- **F** – Frequency

  ```sql
  RUNSTATS TABLESPACE ...TABLE(name) COLGROUP(col) FREQVAL COUNT int
  ```

- **H** – Histogram

  ```sql
  RUNSTATS TABLESPACE ...TABLE(table) COLGROUP(col) HISTOGRAM
  ```
**REASON for Statistics Collection**

- **BASIC**
  - Basic table or index statistics are missing - only use default values

- **CONFLICT**
  - Between table and index statistics or frequency and cardinality statistics
  - Implies statistics were run on different objects at different times

- **LOWCARD**
  - Cardinality of column is a low value, indicates data skew is likely

- **NULLABLE**
  - Distribution statistics not available for a nullable column

- **DEFAULT**
  - Predicate references a value that is probably a default value

- **RANGEPRD**
  - Histogram statistics are not available for a range predicate

- **PARALLEL**
  - Parallelism improved by uniform partitioning of key ranges

- **COMPFFIX**
  - Multi-column cardinality statistics needed for index compound filter factor

*Just recommendations...
No guarantee it’s really necessary*
RID Processing Extension of Workfile Usage

- Prior to 11
  - Terminations(failures) can occur and result in table space scan
    - Many cases addressed in DB2 10 with work file usage for RID overflow

<table>
<thead>
<tr>
<th>RID POOL</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX BLOCKS ALLOCATED</td>
<td>---------</td>
</tr>
<tr>
<td>CURRENT BLKS ALLOCATED</td>
<td>---------</td>
</tr>
<tr>
<td>FAILED - NO STORAGE</td>
<td></td>
</tr>
<tr>
<td>FAILED - RDS LIMIT</td>
<td></td>
</tr>
<tr>
<td>FAILED - DM LIMIT</td>
<td></td>
</tr>
<tr>
<td>FAILED - PROCESS LIMIT</td>
<td></td>
</tr>
</tbody>
</table>

If this is a non-zero value, the RID pool is in trouble and is undersized

Indicates that >25% of the RIDs in the index have qualified

Indicates that you have selected over 28 million RIDs

Causes:
- Inaccurate or incomplete (no frequency or distribution) statistics
- Use of LIKE
- Use of host variables or parameter markers for range predicates (BETWEEN, >, <)

- DB2 11
  - RID overflow to work file can be used for aggregate functions
    - SUM, MAX, MIN, AVG, COUNT
  - Can still be expensive and should be limited - MAXTEMP_RIDS
Stage 2 Predicates Pushdown – List Prefetch Support

- Prior DB2 11
  - In DB2 10 some stage 2 predicates can be processed during stage 1
    - Evaluated by the Index Manager
  - Call made from stage 1 to stage 2
    - Data can be eliminated earlier in the process and indexes can be used
  - Predicate pushdown applies to these couple of examples:
    - Basic predicate (COL op value), BETWEEN, NULL predicate
    - Some expressions and built-in scalar functions(i.e. SUBSTR)
- DB2 11
  - Stage 2 predicate pushdown can occur for list prefetch access (single index)

<table>
<thead>
<tr>
<th>QUERYNO</th>
<th>PREDNO</th>
<th>STAGE</th>
<th>PUSHDOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>2</td>
<td>MATCHING</td>
<td></td>
</tr>
<tr>
<td>1234</td>
<td>3</td>
<td>STAGE2</td>
<td>I</td>
</tr>
</tbody>
</table>

I - Index Manager evaluates predicate
D - Data Manager evaluates predicate
blank – no push down
Conversion of Stage 2 Predicates to Indexable

- Prior to 11
  - Several stage 2 predicates often used were still not indexable
- DB2 11 (CM after Rebind)
  - More stage 2 predicates are converted to be indexable
    - Queries are rewritten

```
DATE()
YEAR()
SUBSTR(col,1,n)
value BETWEEN COL1 AND COL2
```

- If an index on expression exists, conversion will not occur
- Works with literals, host variables, parameter markers
- Works with EQUAL, IN, BETWEEN, and range predicates

```
SELECT COL1, COL2
FROM TAB1
WHERE :hv BETWEEN COL1 AND COL2

SELECT COL1, COL2
FROM TAB`
WHERE COL1 <= :hv AND COL2 >= :hv
```

Not-indexable

Indexable

© YL&A 1999-2015
Additional Indexable Predicates

- Prior to DB2 11
  - CASE expressions were processed at stage 2
  - OR/IN and OR COL IS NULL predicates stage 2
- DB2 11 (CM after rebind)
  - CASE expressions are now indexable
    - For local predicates and join predicates (when CASE evaluated first)

```
SELECT COL1 FROM TAB1, TAB2
WHERE TAB2.COL1 = CASE
  WHEN TAB1.COL2 = 'XYZ' THEN ELSE END
```

- Improved single matching index access for

```
OR COL1 IS NULL \[\rightarrow\] WHERE COL1 = ? OR COL1 IS NULL
```

- Multi-index access allowed for IN/OR

```
WHERE COL1 = ? OR COL2 IN (1,2) \[\rightarrow\] WHERE COL1 = ?
  OR COL2 = 1 OR COL2 = 2
```
**View and Table Expression Predicate Pushdown**

- **Issue**
  - Materialized view and table expressions can be expensive
  - Need to be able to push predicates into view or table expression
- **DB2 11**
  - Provides additional pushdown into views and table expression
    - Non-boolean term (OR) predicate
    - Stage 2 predicates (expressions)
    - Outer join predicate in ON clause
    - Scalar function in SELECT list of view or table expression

```
SELECT COL1, COL2, CNT
FROM TABL1 A,
(SELECT COL3, COUNT(*)
FROM TAB1
GROUP BY COL3) AS B(COL3, CNT)
WHERE A.COL2 = B.COL3
AND SUBSTR( B.COL3, 1, 4) = 'ABCD'
```

Will be pushed down into table expression
Correlated to Non-Correlated Rewrite - More Opportunities

- **DB2 10**
  - Optimizer can rewrite correlated subqueries to be non-correlated
    - If correlation predicates are covered by local predicates in outer query
    - May potentially result in additional index matching predicate
  - Did not apply to
    - Non-boolean term predicates
    - UNION
    - UNION ALL
    - Mismatch data types and lengths

- **DB2 11**
  - Correlated subqueries are converted to non-correlated subqueries
    - When covered by local predicate
    - Including
      - Non-boolean term predicates
      - UNION
      - UNION ALL
      - Mismatch data types and lengths
Correlated to Non-Correlated Rewrite – UNION/UNION ALL

- DB2 11
  - Correlated subqueries are converted to non-correlated subqueries
    - When covered by local predicate
    - Including UNION and UNION ALL term predicates

**UNION ALL**

```
SELECT * FROM TAB1 A
WHERE A.COL1 = 2
AND A.COL2 = (SELECT MAX(B.COL2)
FROM (SELECT MAX(B.COL2)
FROM TAB1 B
UNION ALL
SELECT MAX(TAB2.COL2) FROM TAB2
WHERE TAB2.C1 = 2) AS B );
```

```
SELECT * FROM TAB1 A
WHERE A.COL1 = 2
AND A.COL2 = (SELECT MAX(TAB2.COL2) FROM TAB2
WHERE TAB2.C1 = 2) AS B );
```

All correlated columns replaced with literal constant
Correlated to Non-Correlated Rewrite – Mismatched Types

- DB2 11
  - Correlated subqueries are converted to non-correlated subqueries
    - When covered by local predicate
    - Including predicates with mismatched data types and lengths

Mismatched Data Types

SELECT * FROM TAB1 A
WHERE A.COL1 = 2
AND A.COL2 = (SELECT MAX(B.COL2)
FROM TAB2 B WHERE B.COL1 = A.COL1);

Assuming COL1 is SMALLINT and COL2 is INT

SELECT * FROM TAB1 A
WHERE A.COL1 = 2
AND A.COL2 = (SELECT MAX(B.COL2)
FROM TAB2 B WHERE B.C1 = 2);

Correlated columns replaced with literal constant

Does not apply if A.COL1 is a host variable or parameter marker

Now indexable
In Memory Sort Limit Support

- Prior to DB2 11
  - DB2 use storage for the final in-memory work file storage for final sort
  - For some queries containing ORDER BYs and/or GROUP BYs, this can be a large amount
  - Can consume a large amount of available storage and could not be limited

- DB2 11
  - Can specify maximum allocation of storage to be used
    - For queries containing an ORDER BY, GROUP BY, or both
  - Subsystem parameter MAXSORT_IN_MEMORY
    - Maximum allocation of storage (KB)
  - Only allocated during query processing
  - Used for the final in-memory work file storage for the final sort
  - For queries needing large amounts of real storage when many are running together
    - Could increase this
  - Values: 1000(default) to value of SORT POOL SIZE (whichever is larger)
    - Default SORT POOL is 10,000
Duplicate Removal During Sort Avoidance

Prior to DB2 11

- GROUP BY, DISTINCT, or non-correlated subqueries used an index to ensure order, discarding duplicate entries and avoiding a formal sort
  - Sort was avoided, but every index entry was still processed
  - Leaf pages scanned and duplicates removed before returning data

DB2 11

- GROUP BY, DISTINCT, and non-correlated subquery sort avoidance improved
  - Processes first value within each set of duplicate index values and then on to next value
    - Reduces number of index keys and RIDs processed, and number of leaf pages scanned
  - Uses non-leaf key information to skip forward to find next distinct key
    - Duplicates are removed earlier by using index look aside (leaf high key to non-leaf) and skipping them within the index
      - Regardless of distance between distinct entries
      - Performance benefit when whole index leaf pages can be skipped
  - Reported in DSN_DETCOST_TABLE – IXSCAN_SKIP_DUPS
    - Duplicate index values skipped during index scan, sort avoided
GROUP BY/DISTINCT Early Out for Joins

- Before DB2 11
  - Only correlated EXISTS subqueries that are transformed to a join could take advantage of early-out join processing
- DB2 11
  - Optimization of DISTINCT and other duplicate removal patterns extends to join queries
    - Where join is coded as an existence check
    - Any duplicates introduced from the join are not required for final result
  - Early-out join for inner table can be used
    - As soon as first match is found
    - Rather than processing all matching rows
    - Inner table probe stops after first match is found
    - Also applies to non-Boolean term join conditions with an early-out table
  - Reported in DSN_DETCOST_TABLE – EARLY_OUT
    - Y/N - fetching from table stops after first qualified row
Optimization of Invariant Expressions

- Prior to DB2 11
  - Invariant expressions are expressions whose results do not change during SQL execution
  - Expressions in the SELECT list
    - Are evaluated for each qualified row
- DB2 11
  - Optimized queries with invariant expressions
  - Only evaluated once during the SQL execution

```
SELECT CURRENT DATE + 2 DAYS
```

Executed once
Regardless of number of qualifying rows
Group Bufferpool Write Around

• Issue:
  • GBP dependent objects can have changed pages cached in GBP(GBPCACHE ALL/CHANGED)
  • When batch jobs or utilities run against GBP-dependent objects
    • Can result in heavy GBP page write activity
    • Can result in application slowdowns or pages being written to LPL
  • Large over-allocated GBPs drive up cost
    • Large number of changed pages results in flood of castout CF commands
  • DB2 11(CM)
    • Can bypass writing pages to GBP and write pages directly to disk
      • GBP still sends XI signals
    • Occurs when GBP threshold reaches 50% or class threshold reaches 20%
    • Only pages in GBP via deferred write, asynchronous write, are eligible
      • Not for those from commits or index page splitting

-DISPLAY GROUPBUFFERPOOL(GBP0) MDETAIL
DSNB750I -D1B1 DISPLAY FOR GROUP BUFFER POOL GBP0 FOLLOWS
...
DSNB777I -D1B1 ASYNCHRONOUS WRITES
CHANGED PAGES = 0
CLEAN PAGES = 0
FAILED DUE TO LACK OF STORAGE = 0
WRITE-AROUND PAGES = 0

Number of changed pages written to disk through GBP write-around protocol
GBP Castout – I/O Wait Time and Message Size Reduction

• Prior to DB2 11
  • Heavy write activity cause pages to be written to group buffer pools faster than castout engines can process them
    • GBPs become congested with changed pages
    • May experience full conditions
    • Can result in increased application response time
  • DB2 waits until a page read from GBP was written to disk before another page is read from GBP
  • Notification message sent to castout owners is a list of pages
    • Can be large if many pages are cast out

• DB2 11
  • Reduced wait time for I/O completion
    • Read of GBP for castout processing now overlaps write I/O
  • Reduced notify message size sent to castout owners
    • Size of message indicating status of castout processing is reduced
      – List of page sets or partitions, instead of pages
GBP Class Castout Granularity – Absolute Pages

• Prior to DB2 11
  • Class castout threshold was specified as a percentage of number of data entries
  • Smallest threshold was 1% (data entries per page set)
  • Very large GBPs can have thousands of pages being cast out at a time
    • Stressing castout engines and coupling facility

• DB2 11
  • More granular class castout threshold
  • Class castout threshold can be specified as
    • Percentage of number of data entries
    • or absolute number of pages
  • New syntax for CLASST option

```
CLASST(class-threshold1,class-threshold2)
class-threshold1 - % of data entries
  0 - 90 (default 5)
class-threshold2 - absolute # of pages
  0 - 32767 (default 0)
```

Do not specify value for both
class-threshold2 is ignored if both used

-ALTER GBPOOL(GBP0) CLASST(0,500)

-D1B1 DISPLAY FOR GROUP BUFFER POOL GBP0
...
-D1B1 CLASS CASTOUT THRESHOLD = 0, 500
GROUP BUFFER POOL CASTOUT THRESHOLD = 30%
Partition Level Reorgs and NPIs

- Prior to DB2 11
  - REORG performance degraded due to building shadow NPSIs when Build 2 phase was removed in DB2 9
  - Shadow NPSIs are populated initially with keys of partitions which are not in scope of REORG during UNLOAD phase
    - Keys from parts within scope of REORG are sorted and inserted into shadow NPSI during SORT and REBUILD phases, respectively
  - Performance can be achieved by sorting all keys of NPSI in same sort operation and rebuilding index from entire set of sorted keys
- DB2 11
  - Processing of NPSIs in this case is done in one of the following ways:
    - During UNLOAD, one or more subtasks unload NPSI keys from partitions not within scope of REORG and build shadow NPSI
      - Keys from partitions within scope of REORG are generated from reorganized data rows, sorted, and inserted in shadow index
    - During UNLOAD, subtasks process NPSI keys from partitions not within scope of REORG
      - Keys are routed to sort process to be sorted with keys from partitions within scope of REORG
        - Shadow NPSI is built from this sorted set of keys
        - Can improve performance and leaves previous behavior intact
Partition Level Reorgs and NPIs (cont..)

- Can control behavior
  - `DSNZPARM REORG_PART_SORT_NPSI`
    - Default for sorting non-partitioned index during REORG
    - `AUTO`, `YES`, `NO`
      - `AUTO` - if sorting all keys of NPSIs improves elapsed time and CPU performance, all keys are sorted
        - Uses catalog statistics and RTS
      - `YES` - if sorting all keys of NSPIs improves elapsed time, all keys are sorted
      - `NO` - only keys of NPSIs that are in scope of REORG are sorted

- `SORTNPSI` keyword on REORG
  - `YES` or `AUTO`, all keys can be sorted
  - Not specified, and `REORG_PART_SORT_NPSI` is set to `YES` or `AUTO`, all keys can be sorted as well
  - If either two parameters are set to `NO`, previous method is used

REORG_PART_SORT_NPSI

DB2 10 with PTF UK78229
Physical Deletion of Empty PBG Partitions

• Prior to DB2 11
  • REORG of a partition-by-growth (PBG) table space can result in empty physical partitions at the end
• DB2 11
  • An empty trailing partition occurs when REORG utility moves all data records from a partition into lower numbered partitions
  • New option to remove empty partitions
  • DSNZPARM - REORG_DROP_PBG_PARTS value
    • Whether REORG utility removes trailing empty partitions when operating on an entire PBG table space
    • DISABLE, ENABLE
  • Only applies to a PBG table space REORG
    • It is ignored for other types of table spaces and for REORG of a PBG if you specify PART keyword
REORG without Sorting

- Prior to 11
  - Users do not have ability to avoid cost of re-clustering during a REORG
    - Rows are sorted in sequence of explicit or implicit clustering index
  - SORTDATA NO - DB2 unloads using cluster index or unloading in physical order and pass it to DFSORT
  - Many new reasons to REORG other than reclustering
    - Materialize changes, convert table spaces etc..

- DB2 11
  - Support of SORTDATA NO with SHRLEVEL CHANGE
  - New REORG parameter RECLUSTER YES/NO option on SORTDATA NO
    - RECLUSTER NO - REORG does not unload data through clustering index and does not sort data records in clustering order
  - Can run a REORG without CLUSTERing data
    - Could be a pending change, or data does not need sorting
  - Can now specify SORTDATA NO RECLUSTER NO and DB2 does not sort data
  - Cuts down cost of SORTing data
  - Helps users who use SORTDATA NO but do not have enough DASD space needed by DFSORT to sort data

RECLUSTER YES/NO
RUNSTATS RESET

• Issue:
  • Changes to target objects and many RUNSTATS invocations, with different options, might result in some previously collected statistics becoming outdated
  • May cause DB2 to choose inefficient access paths for SQL statements
    • One solution is to invoke RUNSTATS again to refresh statistics
    • Formulating RUNSTATS may difficult due to previous RUNSTATS executions
• DB2 11
  • Use RUNSTATS to reset access path statistics for all tables/indexes in table space
    • When statistics are reset, default values are used in catalog
      – Some rows are deleted in catalog tables
    • No statistics are gathered or reported
    • Space statistics and real-time statistics are not reset for the specified objects
    • Previous values cannot be recovered
    • Dynamic statement cache invalidated
  • To reset access path statistics with RUNSTATS
    • Specify RESET ACCESSPATH
    • Option HISTORY ACCESSPATH records that access path statistics were reset in rows in SYSTABLES_HIST and SYSINDEXES_HIST
      – Only records that reset occurred, does not save access path statistics values that are reset
Rollback to Savepoint – Performance Improvement

- Prior to DB2 11
  - Savepoints are essentially a placeholder or bookmark in a unit of work
  - Can roll back work to a named savepoint in the unit of work
  - Can experience performance degradation
    - Due to increasing number of log records being scanned with each rollback request
- DB2 11
  - Point in log where previous rollback completed is remembered

```
>>__ROLLBACK__|______|______________________________________________________><
|_TO SAVEPOINT________________|
|_svpt-name_|
```
AUTONOMOUS SQL Procedures

- DB2 11
  - DB2 executes SQL procedure in a unit of work
  - Independent from calling application
  - Follows rules of COMMIT ON RETURN YES before returning to calling application
    - Does not commit changes in calling application
    - Does not impact changes completed by calling application program
  - Calling application program controls when its own updates are committed or rolled back

```sql
> _CREATE PROCEDURE__procedure-name__________>

> _ALTER PROCEDURE__procedure-name__________>

> _PACKAGE OWNER__auth-name__ASUTIME_NO LIMIT__COMMIT ON RETURN_YES__>
    | _ASUTIME_LIMIT__int   | _COMMIT ON RETURN NO__|
    | _AUTONOMOUS__________|
```
XML Performance Improvements

- Insert Improvements (10)
  - Allows randomization of the DOCIDs
  - Eliminates the hotspots in both indexes
- Revalidation avoidance on LOAD
  - If data was already validated according to same schema during initial load or insert
    - Will avoid revalidation - significant elapsed time savings CPU savings
  - Partial validation
    - Provides capability to revalidate only changed part of a document
- XMLTABLE enhancements
  - Remove unreferenced column definitions (10)
  - Merge common column path expressions (10)
  - Storage reuse for output XML columns (10)
  - Date/Time predicate pushdown
  - Optimize index key range for variable character predicates
  - Pushdown of column casting into Xpath
Package Based Continuous Block Fetch

- Prior to DB2 11
  - DDF connections could use SQL-base continuous block fetch

- DB2 11
  - Introduces package-based continuous block fetch
  - Improves performance of retrieval of large read-only result sets for a DB2 for z/OS application accessing a remote DB2 for z/OS server
  - Package-based continuous block fetch
    - Significantly reduces number of messages transmitted from requester to retrieve entire result set
      - More efficient than SQL-based continuous block fetch
Package Based Continuous Block Fetch (cont..)

- Requester opens a secondary connection to DB2 for each read-only cursor
  - DB2 server returns extra query blocks until cursor is exhausted
    - or until application closes cursor
  - When DB2 closes cursor, connection is also closed
- DB2 can send numerous query blocks per request
- A single connection is used for all SQL
- Other SQL, outside of cursors, cannot use connection while cursor driven blocks are using connection

![Diagram of DB2 requester and DB2 server connections]
Database Availability
RELEASE(DEALLOCATE) Thread Interruption

- Issue prior to 11
  - RELEASE(DEALLOCATE)
    - Used often for performance reasons
    - However, for persistent threads the thread may not be deallocated for a long period of time
    - May need to break into these threads to
      - Perform a BIND REPLACE or REBIND PACKAGE
      - Perform online schema changes to tables or indexes accessed by these threads
      - Run an online REORG utility to materialize pending ALTERs
  - Must identify and stop/cancel any active persistent DB2 threads using RELEASE(DEALLOCATE)
DB2 11

- PKGREL_COMMIT online DSNZPARAM
  - Can be used to break into a persistent thread
- YES (default)
  - DB2 can break into persistent threads at COMMIT or ROLLBACK
  - If DB2 detects a BIND REPLACE or REBIND PACKAGE, DDL statement, utility needing to quiesce or invalidate the application’s package
    - Will implicitly deallocate/release package at COMMIT/ROLLBACK
  - No need to identify in advance and stop or cancel any active persistent DB2 threads with RELEASE(DEALLOCATE)
    - Before attempting a BIND REPLACE/REBIND PACKAGE command, schema change or utility associated with those packages
    - Behavior is same as if it was bound RELEASE(COMMIT)
- Not supported for
  - Packages w/OPEN and HELD cursors at time of COMMIT or ROLLBACK
  - Packages bound with KEEPDYNAMIC(YES)
  - When COMMIT or ROLLBACK occurs within a DB2 stored procedure
Cancel DDF Threads

• Prior to DB2 11
  • Cancelling a DDF thread can experience waits

• DB2 11
  • Cancel DDF threads
    • New FORCE option
  • Must issue command without FORCE first
  • For DDF threads only (z/OS 1.13 APAR OA39392)

```sql
>> CANCEL THREAD(token) DDF THREAD(luwid) DUMP LOCAL NOBACKOUT>

FORCED
```

• DRDA SQLCancel() improvements
  • Interrupt even if waiting on locks, executing stored procedures, or statement forwarded to another DB2
Online Partition Limit Key Alter

- Prior to 11
  - When limit keys for a range-partitioned table space are altered
    - All affected partitions are placed in reorg pending status
    - Partitions need to be loaded with LOAD REPLACE or reorganized

- DB2 11 (NFM)
  - Alter of limit keys for range-partitioned table done as a deferred alter
    - Similar to many other pending changes
  - Limit key values for affected partitions are not applied immediately
    - Recorded in SYSPENDINGDDL
  - Applies to table controlled UTS PBR
    - Not for index-controlled – ALTER would set REORP

PREVENT_ALTERTB_LIMITKEY
Disables ALTERing of limit key values for index-controlled partitioned table spaces

PREVENT_NEW_IXCTRL_PART (DB2 10)
Prevent creation of new index-controlled partitioned tables
Online DROP Column

- Prior to DB2 11 Issue:
  - Removing a column from a table requires an outage
  - Unload/Drop/Recreated/Reload
    - Dependency issues too (views, security..etc)
  - No easy way to remove
- DB2 11(NFM)
  - Can now easily get rid of unused columns
    - Overhead (storage, processing..) to carry unused columns in a table
  - Can drop column using
  - ALTER TABLE ... DROP COLUMN SQL statement (1 column only)
  - Can use option RESTRICT to prevent if dependent objects exists
  - Views are implicitly regenerated
  - Dependent packages are invalidated
  - Dependent cached dynamic SQL invalidated
  - Definition removed from catalog
  - Table must be UTS
  - May impact some utilities

ALTER TABLE ...
DROP COLUMN col1
RESTRICT

AREOR status set - Need online REORG CHANGE/REFERENCE to materialize

Cannot be dropped if any views, indexes, unique constraints, or referential constraints are dependent
**DROP COLUMN Restrictions**

- Table space is not a UTS
- Table is
  - a created global temporary table
  - a system-period temporal table
  - a history table
  - an archive-enabled table
  - an archive table
  - an edit proc or validation exit proc
  - defined with a check constraint
  - An MQT
  - referenced in an MQT

- Column is
  - defined as a security label column
  - an XML column
  - a DOCID column
  - a hidden ROWID column
  - as ROWID GENERATED BY DEFAULT, and table has a hidden ROWID column
  - a ROWID column with a dependent LOB column
  - part of table partitioning key
  - part of hash key

- All remaining columns in table are hidden
- A view that is dependent on table has INSTEAD OF triggers
- A trigger is defined on table
- Dependent objects on table
  - Extended indexes, row permissions or column masks, inline SQL table functions
Deferred Definition Enhancement

- Prior to DB2 11
  - A single DBD could contain several objects created with DEFINE(NO)
  - During first insert/LOAD
    - Could experience lock time out on DBD
    - DBD lock is not released until commit
    - Long running UR holds lock
      - Other URs will wait
  
- DB2 11
  - Release DBD lock
    - When data set is created and catalog is updated
    - Before UR commits
  - PM80967 retro-fit for DB2 10
PIT Recovery for Deferred Schema Changes

- **DB2 10**
  - Deferred online schema changes require a REORG to be materialized
  - Does not allow point-in-time (PIT) recoveries
    - After materializing REORG is performed
- **DB2 11(NFM)**
  - Restriction lifted for some pending alters
  - PIT recoveries are now possible after a materializing REORG
**RESTART LIGHT (CASTOUT)**

- **Prior to DB2 11**
  - Restart light released most, but not all, retained locks
  - However, restart light did not go through castout processing
    - Retained page set P-locks in IX or SIX mode were not released
  - Utilities can be blocked from running by these retained page set P-locks
    - Impacting overall DB2 availability

- **DB2 11**
  - New LIGHT (CASTOUT) option
  - Causes all retained locks to be removed
    - Except in-doubt or postponed abort URs
  - Accomplished by initiating castout at end of restart light
    - Pagesets now non-GBP-dependent and retained P-locks released
  - Utilities can now be run after restart light completes

```plaintext
>>__START DB2__ ________________________________________________________>
>__ _____________________ __ __________________ ________________________>
|          _*_____    |  |         _NO_________    |
|_ACCESS(_|_MAINT_|_)_|  |_LIGHT(_|_YES________|_)_|
|_NOINDOUBTS_|
|_CASTOUT____|
```
Summary
Summary – DB2 11 Performance and Availability

- DB2 11 for z/OS offers many opportunities for improving performance and increasing availability
  - Database Features
    - No log declared temps
    - Index pseudo-delete cleanup
    - Suppress-null indexes
  - SQL and Application Features
    - Improved access paths
    - Deallocate for distributed threads
    - More indexable predicates
    - XML update/insert improvements
  - System Features
    - Buffer pool improvements
    - DDF enhancements
    - Work file enhancements
- Several new features are designed to provide better performance
  - Must consider what it may take to implement and future maintainability
  - Must consider true usage capabilities
Courses by YL&A

- DB2 11 for z/OS Transition
  - Application or DBA or both
- DB2 10 for z/OS Transition
  - Application or DBA or both
- DB2 High Performance Design and Tuning
  - Application, Database, Systems
- DB2 Data Sharing
  - Implementation, Performance, Recovery
- DB2 11 for z/OS DBA Certification Crammer
  - Covers exam 610 and 530
- DB2 10 for z/OS DBA Certification Crammer
  - Covers exam 610 and 612
- Application Development and SQL
  - Design, Tuning and Performance

All classes are customized based upon customer requirements

info@ylassoc.com
CPU Reduction Through Performance Audits

- DB2 Performance Audits
  - Existing or new database designs and applications
  - Certification of design and implementation acceptance
  - Evaluation of all the performance ‘points’ in a DB2 environment
    - Physical Design
    - Subsystem
    - Application Code and SQL
  - Help with bringing legacy application to an e-business environment – the rules have changed!
    - What was acceptable performance in the past is NOT acceptable in today’s environments
  - Experienced in ‘fighting fires’ – many performance problems do not become reality until production
  - Results: problems identified, solutions proposed (many implemented immediately), continual knowledge transfer during the process

Cost Avoidance Through Performance Tuning!!!!