DB2 10 and 11 for z/OS Performance Opportunities

YLA Inc.
3309 Robbins Road PMB 226
Springfield, IL 62704

www.ylassoc.com
info@ylassoc.com

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

This presentation looks at the features of DB2 10 and 11 for z/OS with an emphasis on the features focusing on performance. DB2 10 and 11 promises opportunities for CPU savings in various areas. We will review those enhancements and look at all available features in the database, system and SQL and their potential usage. We will look at where the performance opportunities are, what we need to do to take advantage of them, and any additional considerations when exploiting these features. We will also look at experience with these features and performance realizations.

- Brief discussion of overall performance objectives of DB2 10 and 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 goals and strategies for migration and feature exploitation

Performance 'Opportunities'

- When it comes to achieving the best performance 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 10

- REBINDing in CM
  - Improved performance from new run time
  - SPROCs disabled and puffing required for prior release packages
  - Maximize virtual storage above the bar (threads)
  - Reduce exposure to problems for prior release packages
  - Plan management use (APCOMPARE)
  - New query optimization and runtime enhancements
    - In-list, paging, views/NTEs, predicate pushdown and index probing
  - Runtime improvements
    - RID overflow
    - Query parallelism
  - Need to collect RUNSTATS before
- More improvements in CPU when utilizing new features
  - Inserts (depending on table space type and settings)
  - Update/Delete (depending on ability to reduce number of indexes)
  - More application improvements (with changes)
    - High Performance Distributed Threads
    - Native SQL Procedures
    - Literal Reuse

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 and System Performance

Index Updates During Insert Using Parallel I/O

INDEX_IO_PARALLELISM

- DB2 10 (CM)
  - Insert into multiple indexes on same table in parallel
  - Index insert I/O parallelism manages concurrent I/O requests on different indexes into buffer pool in parallel
    - Overlapping synchronous I/O wait time for indexes
    - Still one processing task
  - Significantly improves performance of I/O bound insert workloads
  - Reduce elapsed time for LOAD RESUME YES SHRLEVEL CHANGE

Index I/O Parallelism - Performance

- Reduces I/O wait for large indexes
  - Only if index is NOT memory resident
- Reduces insert elapsed time and class 2 CPU time
  - Savings are greatest when I/O response times are high
- Small CPU overhead due to additional scheduling of sequential prefetch
  - DBM1 SRB time increases (zIIP eligible)
  - Total CPU time increases
- IFCID 357 and IFCID 358
  - Start and end of index I/O parallel insert processing
  - Monitor for each table insert operation
    - Degree of I/O parallelism
    - Number of synchronous I/O waits avoided

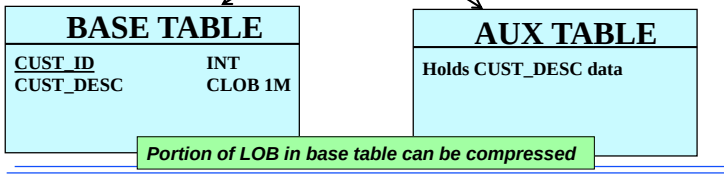
Inline LOBs for Reduced LOB Access Times

- In DB2 10
  - A portion of LOBs can be stored "inline" with the base table data
  - Additional I/O's not required for smaller LOBs

```
CREATE TABLE BASE_TABLE
(CUST_ID INT NOT NULL,
 CUST_DESC CLOB 1M INLINE LENGTH 500)
```

```
SELECT CUST_DESC FROM BASE_TABLE WHERE CUST_ID = 100
```

Will result in the reading of data from both tables only if the length of CUST_DESC for CUST_ID = 100 exceeds 500 bytes



Inline LOBs – All or Portion?

- Completely Inline LOBs
  - Reduce DASD space
    - No more one LOB per page, compression (on base table)
  - Can use DEFINE(NO) - data set for the LOB not defined
  - CPU and I/O saving
    - Avoid LOB aux indexes overhead
- Potential impact on SQLs that do not need LOBs
- Can split LOBs between AUX and base table
  - Part of LOB resides in base and other part in LOB (AUX) table space
  - Will incur the cost of both inline and out of line
  - Index on expression can be used for inline portion of LOB
  - Portion in base can be compressed
  - Normally splitting LOBs in not recommended
    - Unless goal I/O savings is not important but rather the
      - Ability to use index on expression on portion of LOB
        - SUBSTR is only function allowed in index expression
        - Can query by index search for a text string within LOB data
      - Ability to enable faster FETCH CONTINUE for LOB searches

Inline LOBs and Extended Indexes

- Traditionally, LOBs cannot be indexed for performance
- Searching text LOBs could benefit from indexing
- Use of Inline LOBs allow you to index LOB columns
  - By indexing the inline portion of the LOB columns
- Can specify LOB columns for indexes on expression
  - Only SUBSTR built-in function allowed
  - If you can search on the in-line piece of LOB columns
    - Do not need to access the LOB table space to locate a specific LOB

```
CREATE TABLE PRODUCTS (DESCRIPT CLOB(1M) INLINE LENGTH 15);
```

```
CREATE UNIQUE INDEX PRODIX1 ON PRODUCTS
(VARCHAR (SUBSTR(DESCRIPT, 1,15))) ;
```

```
SELECT DESCRIPT FROM PRODUCTS WHERE
VARCHAR(SUBSTR(DESCRIPT,1,15)) = 'HOUSEHOLD GOODS'
```


Additional Non-Key Columns in Unique Index

- Prior to DB2 10
  - An index can be defined to enforce uniqueness
    - Additional indexes may be needed to achieve index only access
      - On columns which are not part of the unique index
  - Additional indexes…
    - Higher insert/delete CPU time
      - Approximately 30% for each index
    - Increased storage
    - Maintenance and availability issues
- DB2 10
  - Ability to add non-key columns in a unique index
  - Can help reduce overall number of indexes
  - Potential CPU reduction in insert
    - IF additional index was removed (IF one had been added before!)

```
CREATE UNIQUE INDEX IX1 ON TAB1(COL1,COL2) INCLUDE (COL3)
or
ALTER INDEX IX1 ADD INCLUDE (COL3)
```

INCLUDE Column Query Impacts

- Considerations with INCLUDE columns
  - Index-only scans perform same as traditional indexes
  - If an index exists merely to enforce uniqueness
    - No risk with INCLUDE
  - However INCLUDEing columns can hurt query performance
    - Queries perform better with slim indexes than with fat indexes

Impacts may include:
Accessing larger index
Additional getpages
Additional sync I/O requests
CPU and elapsed time increase

    - A fat index contains extraneous columns that a query does not use
      - Bad for hybrid joins
        - DB2 index probes w/skip sequential access to the leaf pages
      - Extraneous columns might cause issues with dynamic prefetch
        - Resulting in less prefetch I/O and more synchronous I/O
      - Less of a problem for queries that access a single table
    - Index scan to do more getpages
    - Can degrade buffer pool hit ratio for random selects

Suppress Null Indexes – DB2 11

- Issue
  - DB2 must index every data row when creating an index
    - Affects performance and the size of the index
    - It is useful to exclude one or more values from being indexed
      - Such as values that will never be used in a query
        - NULL, blank, and 0
- DB2 11 (NFM)
  - Improves insert performance of NULL entries by having the 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
  - Improved utility CREATE INDEX performance

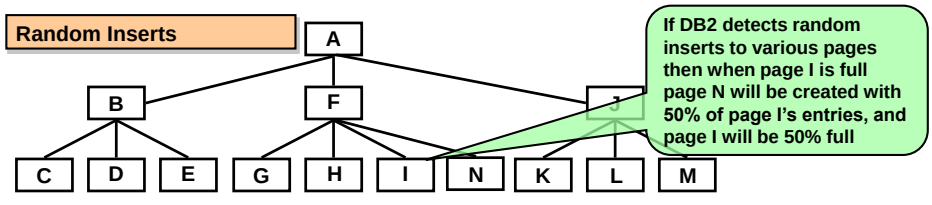
CREATE INDEX…
EXCLUDE NULL KEYS

Compress on-the-fly during INSERT

- DB2 9
  - A REORG or LOAD must occur in order to use DB2 compression
    - Needed to build a compression dictionary
  - LOAD COPYDICTIONARY was introduced to help
    - Allowing for copying of compression dictionary
- DB2 10
  - Tables with COMPRESS YES with no compression dictionary
    - Can be (almost) immediately compressed by DB2
    - Certain operations will trigger creation of compression dictionary
      - INSERT or MERGE
      - LOAD REPLACE, RESUME NO or RESUME YES SHRLEVEL CHANGE
        - Without KEEPDICTIONARY
    - Table space needs to contain enough data(1.2 MB)
    - Compression dictionary is built asynchronously
      - A DB2 service task reads(UR) all tables rows
      - Dictionary is stored in chained page
        - Can span whole table space

Note: Does not work With NOT LOGGED

Index Page Splitting – The issue



- If there is no free page
  - Page splits can become very expensive
  - Very expensive also for GBP dependent indexes in data sharing
- Detection is difficult
  - Performance of inserts begins to suffer
  - Hard to tell when and where the split occurred
- Need better free space
- Maybe need better reorg strategy
- DB2 does not solve the problem
  - Only helps with inserts into the middle of index
  - Does have large page sizes, but this is not answer

IFCID Support for Index Page Splits

- DB2 10
  - Support for IFCID 359
- IFCID 359 records an index page split
  - Makes monitoring index page splits easier
- IFCID 359 stores the following:

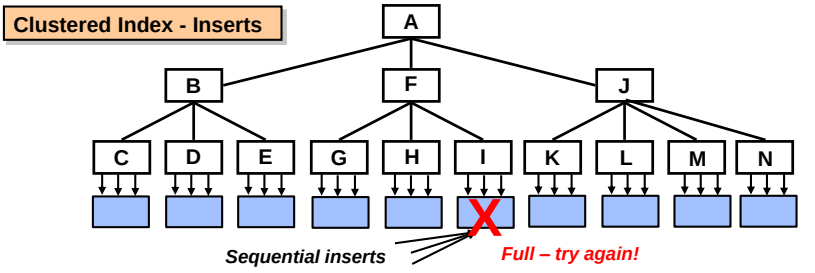
  - Database ID
  - Page set ID
  - Partition number of the splitting index
  - Splitting page number
  - Start timestamp of the split
  - End timestamp of the split

Index Page Splitting – DB2 11

- Prior to DB2 11
  - Index page splitting can result in a great deal of overhead
- DB2 11
  - Improved index split performance
    - Reduces multiple log force write I/Os in data sharing for index split operation
    - Reduces multiple log force write I/Os for pseudo-delete operation
  - Improves index split rollback performance

Sequential Inserts in Clustering Index - Issue

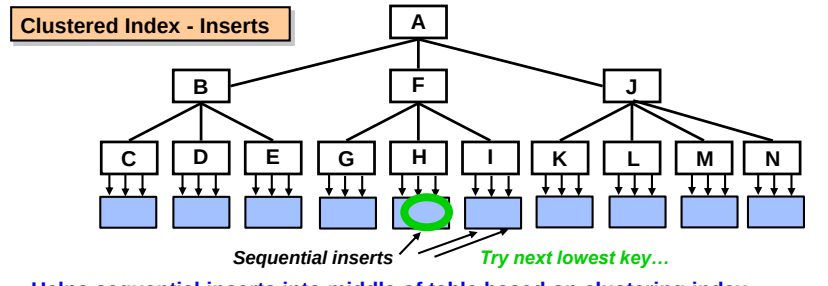
- Prior to DB2 10
  - When a row is inserted
    - DB2 finds candidate page where row is to be inserted according to the clustering index
    - Initial candidate page is data page where row is exist with next highest key



  - If there is available space DB2 inserts row into that page
  - Else, DB2 searches for another candidate page for space to insert row
    - If next row is inserted with key higher than row just inserted but lower than next highest existing row
      - DB2 selects same initial candidate page again (but its full still!)
    - Process is repeated to find another candidate page

Sequential Inserts Improvement in Clustering Index

- In DB2 10 (CM – No rebind)
  - Improves way first candidate page is selected
  - Instead of choosing page pointed to by next highest key as initial candidate page
    - Chooses first candidate page based on next lower key on page where previous row was inserted
      - Chances are that page still contains enough space for new row



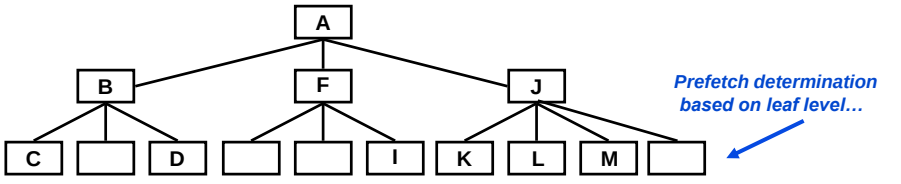
- Helps sequential inserts into middle of table based on clustering index
  - On second/subsequent sequential insert
    - No repetative searching find when first candidate page
    - Provides CPU and getpage savings
      - Fewer candidate pages need to be searched for sequential insert workload

Reduced latch class 006 and 245

Disorganized Indexes - Issue

- Prior to DB2 10
  - A disorganized index results in sequential prefetch failure
    - Due to larger and more frequent gaps between successive leaf pages
      - Large LEAFDIST, LEAFNEAR, LEAFFAR
    - Result:
      - More sync I/O
      - Potentially more index levels
      - Overhead for traversing the index tree

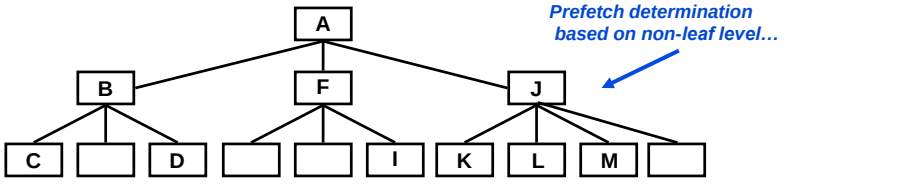
Index with Large LEAFDIST



Disorganized Indexes – Prefetch Toleration

- DB2 10
  - List prefetch can be done for index leaf pages during query processing
    - Based on index non-leaf page information
  - List prefetch usage decision is done at execution
    - Use list prefetch on index leaf pages with range scan
    - Reduces CPU time spent on index root page GETPAGE requests
  - Reduce synchronous I/O waits for queries accessing disorganized indexes
  - Throughput improvement in Reorg, Runstats, Check Index
  - Limited to forward index scan

Index with Large LEAFDIST



Pseudo Deleted Index Entry Cleanup – DB2 11

- Prior to DB2 11
  - Pseudo deleted index entries can cause performance issues
    - Increased getpages, lock requests, CPU
  - Applications may encounter deadlocks and timeouts during update processing
  - REORG INDEX
    - Needed in order to remove pseudo-deleted index entries
- DB2 11
  - Automatically cleans up pseudo deleted entries in indexes
  - zIIP eligible processing runs in the background
    - Minimal or no disruption to applications
  - DSNZPARM to control number of concurrent cleanup tasks(default=10)

INDEX_CLEANUP_THREADS > 0

  - SYSIBM.SYSINDEXCLEANUP catalog table
  - Benefits:
    - Reduce size of some indexes
    - Improve SQL performance, fewer getpages, less CPU and elapsed time
    - Reduce need for REORG INDEX

Help for Reducing Indirect References – DB2 11

- Issue:
  - Updates to variable length and/or compressed rows can increase length of row
  - 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 the original row (RID)
  - RTS indicators REORGNEARINDREF and REORGFARINDREF
    - Used to observe indirect references
  - Indirect references can cause additional I/O
    - To read extra data page into buffer pool
  - REORG TABLESPACE is used to remove indirect references
- DB2 11
  - New PCTFREE FOR UPDATE attribute
    - Used to reserve free space for updates

PCTFREE FOR UPDATE

  - New zparm – PCTFREE_UPD
    - Default = 0
    - Value of -1
      - Autonomic option
      - DB2 uses RTS to determine setting

PCTFREE_UPD

  - SYSTABLEPART – PCTFREE_UPD

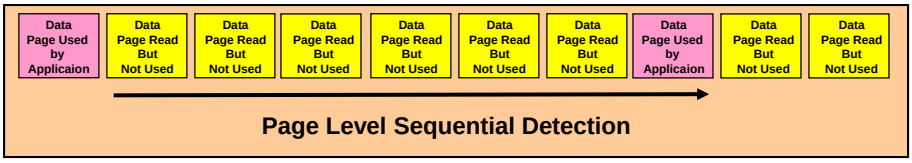
DPSI Improvements - DB2 11

- Improvements to parallelism support for DPSIs
  - Parallelism for joins to DPSIs
    - Parallelism cut on partition/DPSI boundaries
      - On inner table in nested loop join
        - Each child task – 1 partition
      - For outer table
        - Repeated for each child task
        - Maybe use in memory work file if row is less than 32k
  - Straw-model parallelism support for DPSI
    - More tasks than degrees
  - I/O parallelism for single table DPSI access
- Partition elimination on join predicates
  - Only qualified partitions are accessed
  - For a table partitioned on the join columns
    - Each inner table probe only accesses qualified partitions
  - For a table partitioned on non-join columns, and index supporting join is a DPSI
    - Each DPSI partition is processed sequentially

PARAMDEG_DPSI

Sequential Detection - Issue

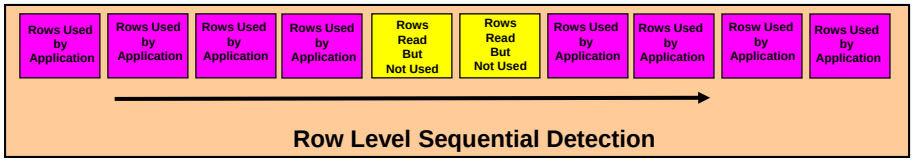
- **Sequential Detection is used if data is accessed repeatedly**
  - **For multiple statements reading forward**
  - **Can be activated by DB2 for data or index leaf pages**
    - **Important for indexes when index lookaside activated**
  - **Most recent 8 pages are tracked**
  - **Tracking is done to determine if page sequential access is happening**
    - **Next page access must be P/2 advancing pages of current page**
      - **P = Prefetch Quantity**
  - **Tracking is continuous**
    - **Can go in and out of sequential detection**



- **Prior to DB2 10**
  - **Dynamic prefetch (based on page level sequential detection) works poorly when number of rows per page is large**

Row Level Sequential Detection

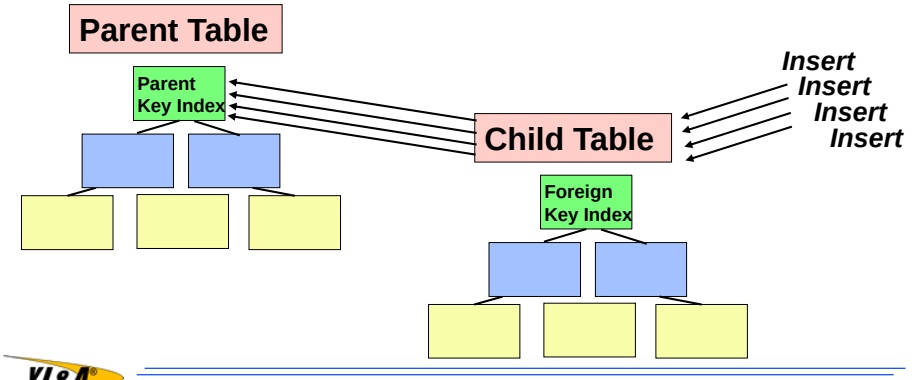
- DB2 10 (CM)
  - Row Level Sequential Detection (RLSD)
  - Count rows, not pages, to track the sequential detection



  - Prefetch will be triggered faster
    - Will use progressive prefetch quantity:
      - For example, with 4K pages the first prefetch I/O reads 8 pages, then 16 pages, then all subsequent I/Os will prefetch 32 pages (as today)
    - Also applies to indexes
      - However, RLSD does not effect indexes
  - RLSD preserves good sequential performance for clustered pages

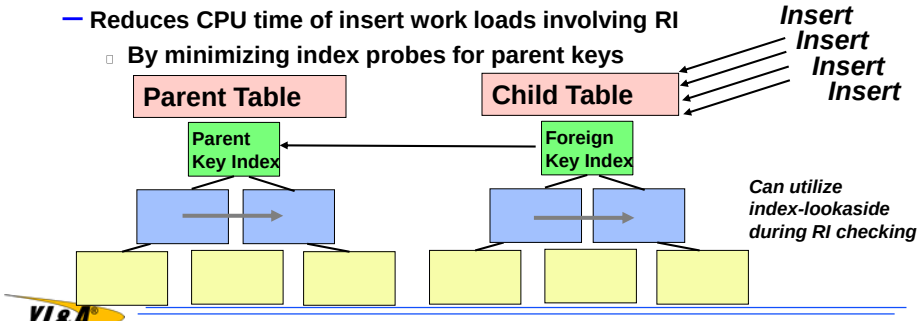
RI Checking During Dependent Inserts

- Prior to DB2 10
  - When inserting into a dependent table
    - DB2 must access the parent key for referential constraint checking
  - No ability to utilize index lookaside or sequential detection for this
  - CPU overhead of RI checking can be excessive
    - Due to repetitive index probes for parent keys
      - Must check to see if parent key exist for new child key



Sequential Detection and Index Lookaside for RI

- DB2 10 (CM)
  - For parent key RI checking DB2 enables
    - Sequential detection for triggering dynamic prefetch
    - Index look-aside
  - Can also avoid index lookup for RI checking
    - If non-unique key to be checked has been checked before
    - 1st INSERT checks parent index, subsequent INSERT will not
      - Must be an foreign key index on the child table
      - Else, only benefit is index look-aside on parent table
  - Reduces CPU time of insert work loads involving RI
    - By minimizing index probes for parent keys



MEMBER CLUSTER Support for UTS

- DB2 9
  - UTS does not support MEMBER CLUSTER
- DB2 10
  - MEMBER CLUSTER is compatible with a universal table space
  - Can be ALTERed
    - Once the table space is a UTS
    - Will need a REORG to materialize
  - SYSTABLESPACE
    - MEMBER_CLUSTER column
  - Space map also now covers 10 segments

```
ALTER TABLESPACE TBS1
MEMBER CLUSTER YES
```

Creating In Memory Tables

- Need for in-memory table
  - Frequently access objects can benefit from being pinned in memory
  - Code/reference tables should not be issuing I/Os for checking
    - Can be very expensive
  - Critical indexes
    - Can also benefit from being in memory
- Prior to DB2 10
  - Difficult to ensure a table/index would be placed in memory and remain there
  - Could define a buffer pool with VPSEQT(0) and PGSTEAL(FIFO)
    - VPSEQT(0) – eliminates sequential queue and turns of prefetch
    - PGSTEAL(FIFO) – turns off latching on queue to determine LRU
    - Also define buffer pool to pages to hold the entire object
      - Obtain number of pages needed from catalog (NACTIVE)
      - Account for growth
  - Bigger issue is getting the data efficiently into the buffer pool

PGSTEAL - NONE

- DB2 10
  - Can define a buffer pool to hold object in memory
  - New PGSTEAL option – NONE
  - Will want to have buffer pool large enough to hold entire object
    - Indexes or tables

PGSTEAL=NONE

- Excellent for lookup/code/reference tables and indexes
  - Small tables that have high I/O rates
  - Avoid LRU chain maintenance (similar to FIFO)
  - Avoid unnecessary prefetch (similar to VPSEQT 0)
- IFCID 201/202
  - Information about in memory object usage

In Memory Tables – Loading the Data into Buffer pool

- Data is loaded sequentially into the buffer pool
  - When first getpage request initiated by the SQL statement
    - An asynchronous task is scheduled under DBM1 to prefetch entire page set into buffer pool
    - CPU is charged to DB2
- If open is part of DB2 restart processing, page sets are not prefetched
- Can physically open before the first SQL access

```
-ACCESS DATABASE(DSN10*) SPACENAM(DSN10*) MODE(OPEN)
```

    - And preload all the data into buffer pool(s) before first SQL access
- Buffer pools need to be large enough to fit all pages of all open page sets
  - Else, I/O delays can occur as buffer pool fills up
    - Then DB2 must steal buffers(on a FIFO basis)
- When object is opened and will remain until closed
- Note: the optimizer takes into consideration that the data is preloaded
  - Access paths can be different when accessing in-memory page sets
    - This access path is not attributed to in-memory page in PLAN_TABLE

ACCESS DATABASE Command Improvements – DB2 11

- ACCESS DATABASE
  - Runs under a separate service task
  - Does not cause queueing of other database commands
  - Runs in parallel

```
-ACCESS DATABASE(DSN11*) SPACENAM(DSN11*) MODE(OPEN)
```

Externalizing RTS Statistics – DB2 11

- **Issue**
  - **RTS data is externalized in SYSIBM.SYSTABLESPACESTATS and SYSIBM.SYSINDEXSPACESTATS**
    - **Every 30 minutes (by default)**
      - **Statistics are essentially 15 minutes old (on average) when tables are accessed**
    - **Objects changing often may not be reflecting correct information used by tools(or users) making recommendations based on RTS(i.e. DSNACCOX)**
- **DB2 11**
  - **Provides a way to externalize RTS**
  - **DB2 ACCESS command has 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)

RUNSTATS - KEYCARD

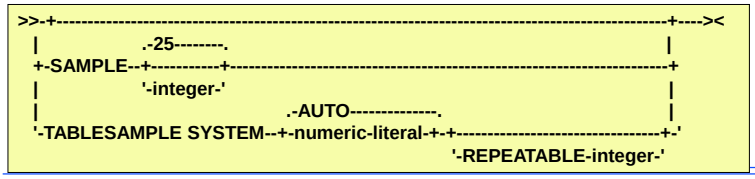
- **Prior to DB2 10**
  - **KEYCARD was optional on RUNSTATS**
- **DB2 10**
  - **KEYCARD is the default**
  - **Run RUNSTATS after REBINDs to 10**
    - **If not collected previously**
  - **Option on RUNSTATS utility is deprecated**

Remind me…
What is KEYCARD???

- **KEYCARD is used in conjunction INDEX on a RUNSTATS utility**
- **Allows RUNSTATS to determine the number of distinct values of combinations of an index's first "n" key columns**
  - **"n" is > than 1 and < than the total number of the index's key columns**
- **Values obtained with KEYCARD populate SYSIBM.SYSCOLDIST**

RUNSTATS – Sampling by Page

- Prior to DB2 10
  - SAMPLE option of RUNSTATS can reduce CPU cost
    - Performs CPU intensive cardinality calculation on a percentage of data
      - However, all 100% of the rows are still read by the RUNSTATS job
      - Works at the row level
- DB2 10
  - TABLESAMPLE option works at page level
    - Provides a reduction in the rows that RUNSTATS will read
    - Can have a significant improvement in performance of RUNSTATS
  - For each page that is included all rows on that page qualify for the sample
  - Every execution of RUNSTATS will usually yield a different sample
    - Unless REPEATABLE clause is specified
  - Only valid on *single-table table spaces* (else defaults to SAMPLE 25)



Statistics Collection/Feedback – DB2 11

- 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
  - More complete and accurate statistics results in more accurate estimates
  - 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(i.e RUNSTATS) can use 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 – DB2 11

- 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 the table contain information used to determine what statistics to collect
  - Can use the 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

Virtual Storage Improvements

- Prior to DB2 10
  - Available virtual storage below the bar limited number of concurrent threads for a single DB2 subsystem
  - DB2 8 moved many DB2 control blocks and work areas (buffer pools, castout buffers, compression dictionaries, RID/EDM pool) above the bar
  - DB2 9 provided additional movement and more DB2 threads
- DB2 10
  - Dramatic reduction of virtual private storage below the bar
    - DBM1 utilization below the 2 GB bar greatly reduced
    - Common storage per thread is reduced by almost by half
  - Majority of current storage moves above the bar
  - More threads per image allowed (approximately 2500)
  - Allows as much as 10x more concurrent active tasks in DB2
  - Less detailed virtual storage monitoring needed
  - Can potentially have fewer DB2 members (less LPARs)
  - Provides cost reductions, simplified management, and easier growth

Additional 10-30% real memory needed

Thread Virtual Storage Changes in DB2 10

- Changes to thread related virtual storage (after rebind in 10)
  - Most control blocks are allocated above-the-bar
  - Plan and package structures allocated above-the-bar
  - Column procedures (SPROC, IPROC, UPROC) are allocated and shared below-the-bar
    - SPROC is no longer limited by 4 KB storage
    - SPROC is enabled for up to 750 columns (previously was 100)
      - Significant CPU reduction when fetching many columns
  - Large fixed areas for EDM thread pools are eliminated with plan and package structures allocated in thread storage pools
  - Stack storage is split between above-the-bar and below-the-bar portions
- Considerations without rebind in 10
  - Plan and package structures allocated below-the-bar
  - Some control blocks are "puffed" at execution below-the-bar for compatible DB2 10 interfaces in case of fallback
  - Column procedures are regenerated for each thread's usage and are not shared among common users of the same statement

Buffer Pool Improvements – DB2 11

- Enhanced sequential vs random classification of pages
  - Dynamic and list prefetch plus sequential format writes
    - Classified as sequential
  - Random getpage which accesses a previously sequentially accessed buffer
    - Reclassified as random
  - More accurate random hit ratio statistics
  - May improve buffer pool hit ratios
- MRU queue management extended to utilities
  - Can be used by utility sequential reads
  - Avoids displacement of useful pages
    - Improving buffer hit ratios
- Real storage page frame size options
  - Larger page frame size improves performance
    - z/OS and EC12 hardware support required
- AUTOSIZE
  - Ability to control the growth of buffer pool using AUTOSIZE
    - Minimum/maximum number of buffers to allocate

FRAMESIZE(2G)

VPSIZEMIN/VPSIZEMAX

Work File Changes

- DB2 10
  - More opportunities for using in-memory work files (CM)
    - Reduced CPU time
      - For queries requiring use of small work files
    - Supports simple predicate evaluation for work files
  - Spanned Pages (NFM)
    - Work file records support larger joins and sorts
      - Ability to span multiple pages
      - Allowing for larger record lengths and larger sort key lengths
        - Max length of a work file record = 65KB
        - Max limit for sort key length = 32KB
  - Utilizes partition-by-growth table spaces (NFM)

Note: there is still cost for every column and row involved a sort!

Declare Temporary Tables and Work Files

- Declared temporary tables still cannot span work files
  - Can be defined work files in a PBG table space
    - Using MAXPARTITIONS, DSSIZE and NUMPARTS to create appropriate sized/expandable work file
    - New partitions will remain through DB2 restart
  - Separation can help alleviate contention between declared temporary tables and other work file usage
  - Ability to expand work files to avoid abends
- SYSTABLESTATS
  - Contains work file data at partition level for the table spaces
- DSNZPARMs allow for soft/hard separation of DGTTs and other work files
  - Separation provided first in APAR in DB2 9
    - Must separate DGTTs work files from other work file users
      - By having secondary's on DGTTs
      - No secondary's on other work files
  - WFDBSEP (default NO)
    - Hard separation – use work files solely for DGTTs and non-DGTT work

WFDBSEP

Work File Enhancement – DB2 11

- Prior to DB2 11
  - Issues with separation of work files
  - DGTTs need to use separate space and can run out
- DB2 11
  - Allow for warning messages to be issued when work file database space usage approaches a critical level
  - Allows ability to monitor space used by DGTTs and other work separately
  - WFSTGUSE_AGENT_THRESHOLD
    - Define the agent-level space-usage alert threshold
      - Online-changeable
    - 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

No Log Declared Global Temporary Table – DB2 11

- 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
  - 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
  - Compatible with DB2 family

DECLARED GLOBAL TEMPORARY TABLE tab1….
NOT LOGGED

RID Processing - Problem

- Prior to DB2 10
  - At run time DB2 builds a RID list for query processing for such processes as list prefetch or multi-index access
  - If the RID pools is full or if the predicate qualifies more that 25% of rows
    - DB2 falls back to a table space scan

```
RID POOL                    QUANTITY
----------                  --------
MAX BLOCKS ALLOCATED
CURRENT BLKS ALLOCATED
FAILED - NO STORAGE
FAILED - RDS LIMIT
FAILED - DM LIMIT
FAILED - PROCESS LIMIT
```

IFCID 125
To find problem SQL or DSN_STATEMENT_CACHE_TABLE (if cached dynamic)

If this is a non-zero value, the RID pool is in trouble and is are 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, >, <)**

RID Processing and Work File Usage

- DB2 10
  - At runtime RID overflows now use a work file to continue processing
    - Avoids fallback to table space scan
  - May increase work file usage
    - MAXTEMPS_RID
      - DSNZPARM for maximum work file usage for each RID list
    - Sort type work file usage (as opposed to DGTT-type)
    - Increase is expected to be small
- Some overhead associated with overflowing RIDs to a work file
  - Can result in an increase in CPU and elapsed time
  - Can be larger if you have a constrained work file buffer pool
  - However, maybe less costly than falling back to a table space scan
  - Consider not qualifying so many rids
    - i.e. better query, index, statistics
      - Resulting in different access path
- Runtime decision
  - Does not effect access path selection
  - Not used for hybrid joins
- DB2 can still fall back to a table space scan
  - If there is not enough work file storage for RID requests

MAXTEMPS_RID

RID Pool default increased from 8MB to 400MB

RID Processing Improvements– DB2 11

- Prior to 11
  - RID processing can be expensive
  - Terminations(failures) can happen in some cases and result in table space scan
    - Most cases addressed in DB2 10 with work file usage for RID overflow
      - But not all cases: such as aggregate function
  - Hybrid joins can also consume a great deal of space in the RID pool
- DB2 11
  - RID overflow to work file used for aggregate functions
    - SUM, MAX, MIN, AVG, COUNT
  - Hybrid Join consumption of RID pool restricted to 80%

Data Sharing – LRSN Assignment Improvement

- Prior to DB2 10
  - LRSN had to be unique on a page
  - Could cause latching contention
  - DB2 9 provides a function called LRSN spin avoidance
    - Allows duplicate LRSN values for consecutive log records on a member
    - Consecutive log records that are written for updates to different pages (i.e. data and index page)
      - Can share the same LRSN value
  - Consecutive log records for same index/data page must still be unique
- DB2 10 NFM
  - LRSN in data sharing does not have to be unique within a page
  - Consecutive log records for inserts to the same data page
    - Can have the same LRSN value
  - Performance improvement for data sharing
    - Especially for high volume environments
      - No wait(spin) to get unique LRSN
    - Helps multi-row insert application performance

Only for INSERT!

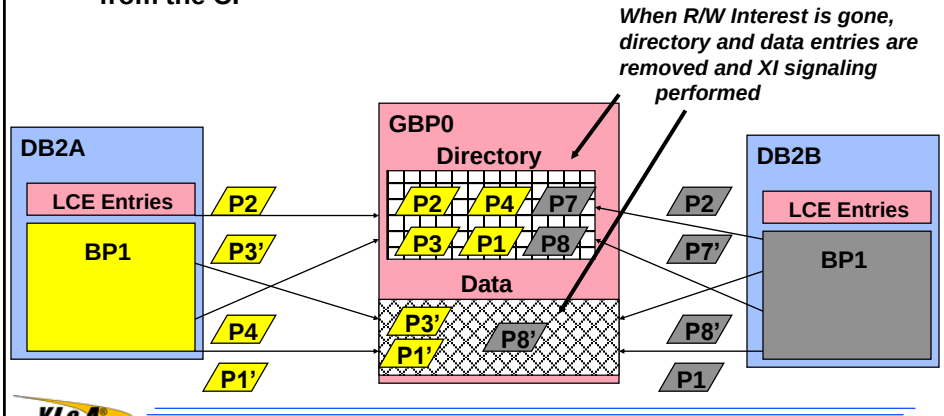
DELETE and UPDATE still require unique LRSN

Extended RBA and LRSN – DB2 11

- DB2 10
  - 6-byte LRSN provides a granularity of 16 microseconds
  - Possibility of many consecutive log records having duplicate LRSN values
    - Considerable overhead
  - Enhancements in 9/10 to reduce spin in the Log Manager to avoid duplicate LRSNs
    - Not 100%
- DB2 11 (NFM)
  - Extends LRSN to use more of the TOD clock precision
  - Eliminates need to spin to obtain a unique value
    - Which improves data sharing performance for batch processes
- RBA and LRSN expanded to 10 bytes
  - 5% greater granularity
  - 30,000 years before the end of range!
    - Will take many decades to exhaust

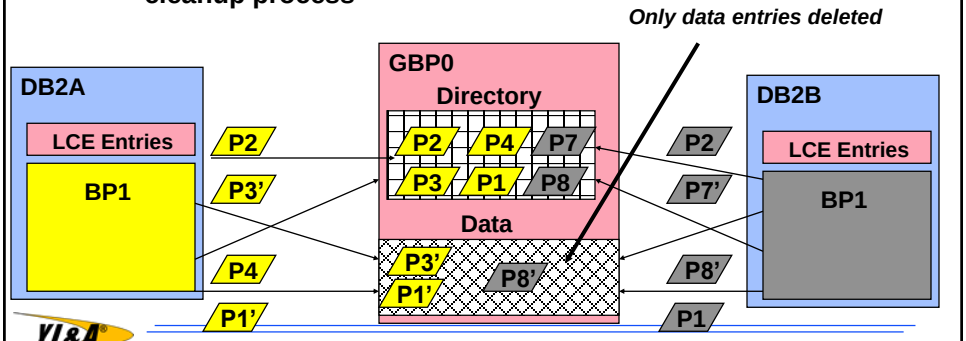
GBP – CF Delete During RW Removal

- Prior to DB2 10
  - IRLM lock timeouts could occur during CF "delete name" requests
    - When removing inter-DB2 Read Write interest
  - Typically associated with a DB2 member physically separated from the CF



Group Buffer Pool Dependency Optimization

- DB2 10
  - When R/W interest is gone, only CF data entries are deleted
  - Avoiding potentially massive amounts XI signaling
  - CF "delete name" happens faster
  - Avoid impact to transactions
  - Directory entry will be eventually be deregistered
    - When local buffer pool pages are reclaimed via a "lazy" cleanup process



GBP Castout Improvements – DB2 11

- 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 operation
    - Wait time is reduced by overlapping read for castout with write to DASD
  - Reduced notify message size sent to castout owners
    - Size of message indicating status of castout processing is reduced
    - Message includes a list of page sets or partitions, instead of a list of pages
      - Considerably reduces size of message.
  - More granular class castout threshold

XML Performance Improvements

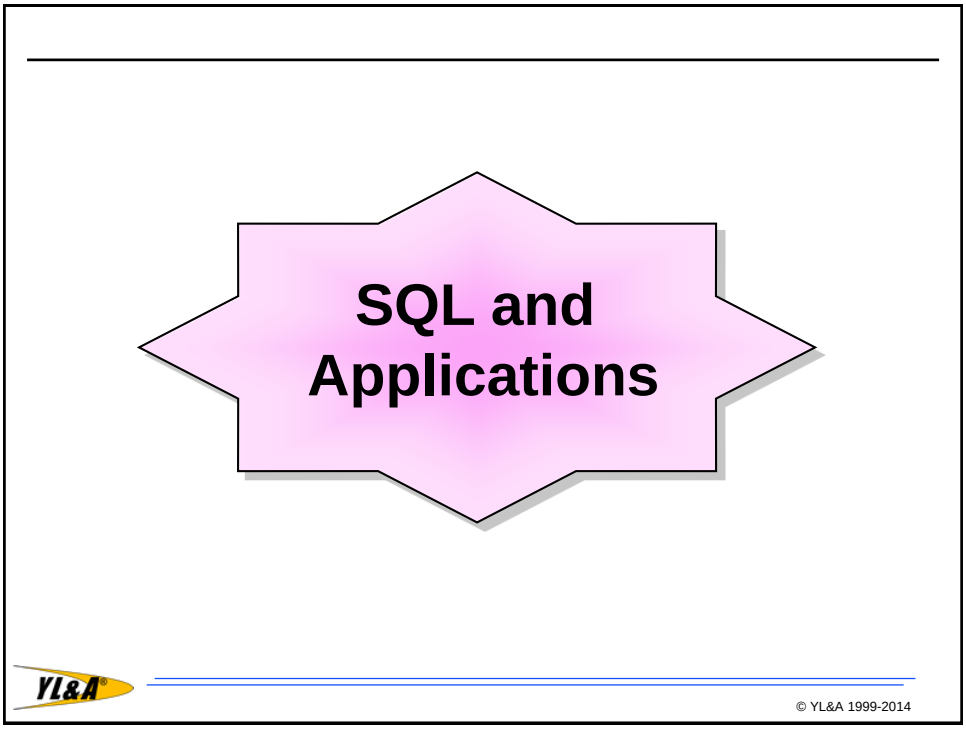
- XML Document Versioning
  - Can lead to improved concurrency through lock avoidance
    - Readers do not need to lock XML documents
      - Improving concurrency for update operations
- XML Update
  - No more full document replacement

```
UPDATE DSN81010.CUSTOMER
SET INFO=XMLMODIFY('replace node /customerinfo/phone with $x',
       XMLPARSE('phone type="work">217-555-1358</phone>') AS "X")
WHERE CID=1000;
```

- Binary XML support
  - Avoid the cost of XML parsing during insert and reduces the size
    - 10-30% CPU and elapsed time improvement
  - DB2 11 – also validates binary XML directly without need to serialize it
- Schema validation in engine (built-in function)
  - No UDF call for validation
  - Utilizes XML System Service Parser
    - 100% zIIP / zAAP eligible

XML Performance Improvements – DB2 11

- 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/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



Parallelism Support – Multi Row Fetch

- **Prior to DB2 10**
  - **Parallelism is disabled for the last parallel group in the top level query block with multi-row fetch**
    - **If there is**
      - **No more tables to join after the parallel group**
      - **No GROUP BY clause**
      - **No ORDER BY clause**
- **DB2 10 (CM)**
  - **Parallelism can be achieved for multi-row fetch for basic queries**
  - **Cursor must be read-only**

SELECT COL1 FROM TABLE

Could not take advantage of parallelism prior to 10
(no parallel group, no ORDER BY, no GROUP BY)

YL&A

Parallelism Degree – Setting Max Degree

- Prior to DB2 10
  - PARAMDEG specifies max degree of parallelism allowed for a parallel group
    - Limits degree of parallelism
      - DB2 cannot create too many parallel tasks using up virtual storage
    - 0(default)value
      - DB2 chooses a max degree of parallelism based on system configuration (max 10x number of CPs)
- DB2 10 (CM)

PARAMDEG

  - 0 = DB2 caps parallelism degree at 2x number of CPs
    - Could experience degradation in query performance(in DB2 10)
      - Due to possibility of a lower degree of parallelism being chosen for some queries
      - May need to set to a higher non-zero value to achieve higher degree of parallelism
  - non-zero = used to cap degree of parallelism
    - Unless degree is provided by an optimization hint
  - Potential to have more CPs in one LPAR due to reduction of DBM1 storage
    - Provides potential for more parallel tasks (if degree cap unchanged)

Dynamic SQL Cache and Literals

- Prior to DB2 10
  - To allow for reuse in cache
    - SQL string had to be identical and SQL executed by same user
    - Programs needed to use parameter markers
      - Similar to using host variables in static SQL
      - SQL then would always be identical
        - Even if values in the parameter markers changed
    - Some dynamic SQL uses literals rather than using parameter markers
      - Literals are likely to be different with every execution of the SQL
        - No reuse of the SQL in cache
        - Prepare must take place for each unique piece of SQL
- DB2 10
  - Literal replacement implementation
    - Dynamic SQL with literals can now be re-used in cache
    - Literals replaced with '&'
      - Similar to parameter markers

Literal Replacement

- Original SQL with literals is looked up in the cache
  - If not found, literals are replaced and new SQL is looked up in the cache
  - Additional match on literal usability
- Can only match with SQL stored with same attribute, not parameter marker
  - If not found, new SQL is prepared and stored in cache
- Reuse of prepared statements in the dynamic statement cache
  - To be eligible for reuse of constant in new and cached statement
    - Text and auth iD
    - Immediate context
    - Data type
    - Data type length and size

    Must be the same
- If both instances meet the criteria for reuse
  - And statement is prepared with CONCENTRATE STATEMENTS WITH LITERALS
    - Can be shared by same SQL statement with different constants
  - New dynamic SQL statement will share the cached statement
    - New statement will use it's own literal constants during execution
      - Not constants of cached statement

Optimization Hints Usage Challenge

- Prior to DB2 10
  - DB2 had user level optimization hints
  - OPTHINTS associated with a query number
    - Poses many challenges
      - For static SQL
        - If code is changed, the precompiler will generate a new query number for each SQL statement occurring after the code change
        - QUERYNO clause could be added to an SQL statement to ensure the same query number is used across application changes
      - For dynamic SQL
        - Adding QUERYNO to SQL statement was required way to match a statement to a hint
    - Either for static or dynamic
      - Altering the SQL statement is often impractical

Statement Level Optimization Hints

- DB2 10 - Statement level optimization hints
  - Alternative way to associate a statement and a hint – using query text
  - Similar to concept of a dynamic statement cache text match
    - SQL text is tied to hint - a static bind or dynamic prepare will attempt to lookup statement text to find a matching hint
    - Hint created irrespective of its usage for dynamic or static SQL
      - Can be given a global scope or package level scope
- Hints are stored in the access path repository
  - PLAN_TABLE is still used
  - Alternate method for looking up the hint
    - Easier for dynamic and more stable for application changes in static SQL
- By tying hint to the statement text
  - Changes in query numbers will not affect the hint
  - Changes to the SQL statement will result in the match failing
- Statement level hints
  - BIND QUERY for static or dynamic SQL
    - Enforce an existing access path from a PLAN_TABLE (hints)
    - Customize optimization options for a statement

Access Path Repository

- The access path repository holds important query metadata
  - Query text and access paths, optimization options, etc..
- Implemented in catalog during ENFM
  - SYSIBM.SYSQUERY
  - SYSIBM.SYSQUERYPLAN
  - SYSIBM.SYSQUERYOPTS
- Input table – userid. DSN_USERQUERY_TABLE
  - SYSQUERY
    - Data(queries) for all optimization hints for each static or dynamic SQL query that is to exploit user-specified hints/options
  - SYSQUERYPLAN
    - Data for optimization hints to enforce an access path for query in SYSQUERY
  - SYSQUERYOPTS
    - Contains data for hints that customize optimization parameters
- Hints and options are mutually exclusive
  - It is only possible to have either a hint or options for a given statement in SYSQUERY

DSN_USERQUERY_TABLE

- DSN_USERQUERY_TABLE
  - QUERYNO
    - Match a PLAN_TABLE QUERYNO(same AuthID) to enforce access path
    - Use QUERYNO that does NOT match a PLAN_TABLE QUERYNO to set optimization parameters
  - HINT_SCOPE
    - 0 - System level access plan hint (matching on statement and schema)
    - 1 - Package level access plan hint (matching on COLL, PKGE, VER)
  - QUERY_TEXT
    - SELECT, INSERT, DELETE, UPDATE, MERGE,TRUNCATE
  - USERFILTER
    - Filter name that groups a set of queries together, or blank
  - COLLECTION
    - Collection of package from SYSIBM.SYSPACKAGE(optional if HINT_SCOPE is 0)
  - PACKAGE
    - Name of package from SYSIBM.SYSPACKAGE(optional if HINT_SCOPE is 0)

Overriding Predicate Selectivity – DB2 11

- Issue
  - Optimizer is always attempting to obtain the lowest cost access path
  - Filtering by predicate is often hard to determine
    - Too many unknowns: skew, bad statistics, host variables, etc..
  - Can result in poor access path selection
- DB2 11
  - Can override selectivity of predicates for matching statements
  - Predicate selectivity overrides
    - Specify selectivity values to be used instead of default filter factors during access path selection
  - Selectivity values apply to all matching statements in the specified context
  - Good approach for statements containing predicates whose selectivities are difficult or impossible for DB2 to estimate
  - DB2 can use improved selectivity information to choose an optimal access path for the SQL statement
    - Other methods(OPTHINT) enforce a particular access path and remove DB2 query optimization from the process
    - Overrides simply provide additional information to the optimizer
  - Selectivity Profile used for filter factor stats collection for predicates
    - Populated by a BIND QUERY

"Filter Factor Hints"

DSN_PREDICATE_SELECTIVITY Table – DB2 11

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

DSN_PREDICATE_SELECTIVITY

INSTANCE	WEIGHT	PREDNO	SELECTIVITY
1	0.45	1	0.26
1	0.45	2	0.10
1	0.45	5	0.10
2	0.25	1	0.99
2	0.25	2	0.25
2	0.25	5	0.01

DSN_USERQUERY_TABLE – Selectivity Override – DB2 11

- SELECTVTY_OVERRIDE
  - Y = indicates the row creates a selectivity override for the query
  - N = no override created
  - Must be set to 'N' when value of ACCESSPATH_HINT is 'Y' or blank, or value of OPTION_OVERRIDE is blank
- ACCESSPATH_HINT
  - Y = an access path for the query is specified
  - N = row does not create a statement-level access path
    - Access paths and selectivity overrides cannot be specified for the same statements
    - Must be N, if either of the values of the SELECTVTY_OVERRIDE or OPTION_OVERRIDE columns are 'Y'
  - Blank = Access path might be specified, if the related PLAN_TABLE instance contains rows that specify the access path.
    - When blank, the value of the OPTION_OVERRIDE column must also be blank and the value of the SELECTIVITY_OVERRIDE column must be 'N'
- OPTION_OVERRIDE
  - Y = create option overrides
  - Unless you specifically want to enable both option overrides and selectivity overrides for the same statement, specify 'N' in this column.

Plan Management

- DB2 10
  - Managed in new catalog table SYSPACKCOPY
    - Information is copied into SYSPACKCOPY during ENFM
    - COPYID column
      - 1 - Previous copy
      - 2 - Original copy
    - Similar data to SYSPACKAGE
- PLANMGMT - DSNZPARM
  - EXTENDED (Default)
    - 3 copies are kept by default
- REBIND option for duplicates
  - APRETAINDUP
    - YES (default)
      - All copies were retained
    - NO
      - New access path identical to a current access path does not get copied to a previous or original

PLANMGMT

APCOMPARE and APRETAINDUP

- APRETAINDUP
  - Avoid storing package copies identical to previous access path
- APCOMPARE
  - Shows id new access paths are different from older access paths

```
REBIND PACKAGE (SRRA.MVBRV41.(2011-06-21-19.23.51.223650) ) PLANMGMT(BASIC)
APCOMPARE(WARN)    APRETAINDUP(NO)
DSNT285I  #DBA1 DSNTBBP2 REBIND FOR PACKAGE = DBA1.SRRA.MVBRV41,
          USE OF APCOMPARE RESULTS IN:
          2 STATEMENTS WHERE COMPARISON IS SUCCESSFUL
          0 STATEMENTS WHERE COMPARISON IS NOT SUCCESSFUL
          0 STATEMENTS WHERE COMPARISON COULD NOT BE PERFORMED.

DSNT288I  #DBA1 DSNTBRB2 REBIND PACKAGE DID NOT PRESERVE THE
          PREVIOUS AND/OR ORIGINAL ACCESS PATHS FOR
          STATIC SQL STATEMENTS IN PACKAGE =
          DBA1.SRRA.MVBRV41.(2011-06-21-19.23.51.223650)
          BECAUSE THEY WERE IDENTICAL TO THE NEW ACCESS PATHS
.
```

ACOMPARE Output – Changed Access Path

```
REBIND PACKAGE (SRRA.MVBRV42.(2011-07-22-18.21.41.212357) ) PLANMGMT(BASIC)
APCOMPARE(WARN)    APRETAINDUP(NO)
```

```
DSNT285I  #DBA1 DSNTBBP2 REBIND FOR PACKAGE = DBA1.SRRA.MVBRV42,
          USE OF APCOMPARE RESULTS IN:
          0 STATEMENTS WHERE COMPARISON IS SUCCESSFUL
          2 STATEMENTS WHERE COMPARISON IS NOT SUCCESSFUL
          0 STATEMENTS WHERE COMPARISON COULD NOT BE PERFORMED.
```

Shows a change exist

- PLAN_TABLE(REMARKS)

APCOMPARE FAILURE (COLUMN: ACCESSTYPE)

- Access Path comparison failures are noted
  - Column in PLAN_TABLE where change is detected is shown

APREUSE

- Access Path Reuse
  - Perform BIND/REBIND
  - Avoid access path changes
    - At migration from DB2 9 to 10
    - After service fixes requiring the regeneration of runtime structures
    - To bring invalid packages back to life
    - Due to application changes (BIND PACKAGE)
- Option on BIND PACKAGE REBIND PACKAGE, REBIND TRIGGER PACKAGE, REBIND for native SQL procedures
  - For all statements in package
  - Loads old access path
  - Feed as hint to the optimizer
  - Compares old/new access paths
- Report results in PLAN_TABLE
  - HINT_USED = APRREUSE
- Determine REBIND success vs failure
  - APREUSE(ERROR)
  - Some access paths may not be able to be reused
- Implicitly turns on APCOMPARE

DB2 11 - APREUSE(WARN)
If failure of reuse
- Optimizer generates new access path
- Failure of one SQL will not fail entire package

EXPLAIN ONLY and EXPLAIN PACKAGE

- Bind/Rebind package EXPLAIN(ONLY)
  - Bind/rebind is performed
  - Performs explain on SQL
  - Access path is chosen
  - Rolled back

EXPLAIN(ONLY)

- Existing package copies are not overwritten
- Externalized in PLAN_TABLE
  - BIND_EXPLAIN_ONLY='Y'
- EXPLAIN PACKAGE command
  - To retrieve PLAN_TABLE records for existing package
  - Original bind/rebind may have been EXPLAIN(NO)
  - Must have been bound in DB2 9 or 10
  - Populate explain tables without BIND or REBIND
- Is really just an extract of Explain information
  - Can specify either the current, previous or original

```
>>------EXPLAIN----PACKAGE-------------------------------------------------------->
>>-----COLLECTION--collection-name--PACKAGE--package-name------>
        +---VERSION-version-name---+ +---COPY--copy-id---+
```


SET CURRENT EXPLAIN MODE for Dynamic SQL

- SET CURRENT EXPLAIN MODE
  - YES (NO by default)
    - Explainable SQL statements run normally
    - Explain information captured after each statement is prepared/executed
  - EXPLAIN
    - Explainable SQL statements do not execute
    - Explain information is captured during prepare
    - Prepared statements not written to dynamic statement cache
      - Gathers access path and run time statistics, no caching/performance opt
      - Not to be used production environment
        - Use wisely when needed
        - Can generate large amounts of EXPLAIN data
        - Every prepare or open/execute performed could be EXPLAINed
- Populates PLAN_TABLE and DSN_STATEMENT_CACHE_TABLE

```
>>__SET CURRENT EXPLAIN MODE =__NO_____________><
                              |_YES___________|
                              |_EXPLAIN_______|
                                |_host-variable_|
```

Performed with EXPLAIN privilege or SQLADM authority

Explain Table Additions – DB2 11

- DSN_DETCOST_TABLE
  - IXSCAN_SKIP_SCREEN
    - Whether key ranges disqualified by index screening predicates are skipped during an index scan
  - IXSCAN_SKIP_DUPS
    - If duplicate index values were skipped during index scan and sort was avoided
  - EARLY_OUT (Y/N)
    - Whether fetching from the table stops after the first qualified row
- DSN_PREDICAT_TABLE (ADDED_PRED)
  - T - Transitive closure
  - B - Bubbleup
  - P - Pushdown
  - L - Localization
  - C - Correlation
  - J - Join
  - K - Like for IOE
  - S - Simplification
  - ' ' Predicate not added by DB2
- DSN_QUERY_TABLE
  - Before and after SQL shown for XML queries

Predicate Transitive Closure Improvements

- Prior to DB2 10
  - Predicate transitive closure supported for the following
    - =, BETWEEN, <, <=, >, >=
- DB2 10 (CM)
  - Predicate transitive closure supported for IN Lists
  - IN Lists can be generated for columns that are also referenced by another predicate

```
SELECT COL1
FROM TAB1, TAB2
WHERE TAB1.COL1 = TAB2.COL1
AND TAB1.COL1 IN ('A', 'B', 'C')
```

```
SELECT COL1
FROM TAB1, TAB2
WHERE TAB1.COL1 = TAB2.COL1
AND TAB1.COL1 IN ('A', 'B', 'C')
AND TAB2.COL1 IN ('A', 'B', 'C')
```

  - With the additional predicate
    - Optimizer may choose to access table in a different order
    - Usually the table that qualifies fewer rows first

IN List Direct Table Access and List Prefetch

- IN List Direct Table Access
  - Uses in-memory tables to process one or more IN-list predicates as matching predicates

```
SELECT COL1
FROM TAB1
WHERE TAB1.COL1 IN ('X', 'Y', 'Z'):
```

  - Supports matching on multiple IN-list predicates
    - If indexes exist on the necessary columns

```
SELECT COL1, COL2
FROM TAB1
WHERE TAB1.COL1 IN ('X', 'Y', 'Z')
AND TAB1.COL2 IN ('P', 'D', 'Q')
```

(X,P),(X,D),(X,Q), (Y,P), (Y,D) etc…

No limit to the number of IN-list predicates that DB2 builds into an in-memory table

  - Takes each of the IN-lists and builds in-memory tables
    - Then uses nested loop joins to join in-memory tables to target table
      - With the potential for matching on multiple columns of the index
- If IN-list predicate is selected as matching predicate and list prefetch is chosen
  - The IN-list predicates are accessed as an in-memory table
  - Otherwise you get normal In-list access without list prefetch

IN List Direct Table Access and List Prefetch (cont..)

- Can be viewed in PLAN_TABLE
  - TNAME = DSNINnnn(QB)
    - nnn=predicate #
    - QB=Query Block
  - TABLE_TYPE= 'I'
  - ACCESSTYPE ='IN'

If IN-list direct table access is not used, then normal access ACCESSTYPE = 'N' with no list prefetch

```
SELECT COL1
FROM TAB1
WHERE TAB1.COL1 IN (?, ?, ?):
```

QBNO	PLANNO	METHOD	TBNAME	ACCESSTYPE	MCLS	ACCESSNAME	TBTYP	PFETCH
1	1	0	DSNIN001(01)	IN	0		I	
1	2	1	TAB1	I	1	IX1	T	L

Shows access to in-list table

Range List Access

- Prior to DB2 10
  - OR predicates often result in multi-index access with list prefetch
  - Not as efficient as single index access
  - Multi-index access retrieves all RIDs that qualify from each OR condition and then unions the result
- DB2 10 (CM)
  - Range-list index scans
    - Processing of OR predicates can be mapped to a single index
    - Improves performance of applications with data-dependent pagination
      - Or in situations where tables hold a denormalization of several different tables providing the same functionality
  - DB2 logically breaks down OR predicate into a range list of two ranges
    - Performs two index probes
    - Scans the index once
      - Consumes fewer RID list resources than multiple index scans
  - Supports single index access without list prefetch

ACCESSTYPE=NR

Range List Access (cont..)

- Requires every OR predicate references the same table and has at least one matching predicate mapped to same index
- Ordering of the access path steps will be determined at run time
  - PLAN_TABLE represents the order coded in the query

Index: LNAME, FNAME

```
SELECT LNAME, FNAME, ADDRESS
FROM CUSTOMER
WHERE
  (LNAME = 'Lawson'  AND FNAME > 'Susan')
   OR (LNAME > 'Lawson')
ORDER BY LNAME, FNAME
FETCH FIRST 5 ROWS ONLY;
```

First probe

(LNAME = 'Lawson' AND FNAME > 'Susan')

All remaining 'Lawson' rows are scanned until no more rows qualify the first probe predicate or until the application stops fetching rows
No need to sort the result set because rows are retrieved in required order through the index

If fetching 5 rows is not satisfied by first probe, then a second probe of the index is executed with (LNAME > 'Lawson')
A new position is established in the index and scanning continues until either all rows are processed or until the required number of rows are returned to the application

Second probe

(LNAME > 'Lawson')

QBNO	PLANNO	METHOD	TBNAME	ACCESSTYPE	MC	ACCESSNAME	QBTYPE	MIXOPSEQ
1	1	0	CUSTOMER	NR	2	IX1	SELECT	1
1	1	0	CUSTOMER	NR	1	IX1	SELECT	2

Stage 2 Predicates Processing Improvement

- **Prior DB2 10**
  - **Stage 1 predicates processed first**
  - **Data is then passed from stage 1 to stage 2**
  - **Stage 2 predicates then processed**
  - **Cannot use an index to reduce data retrieved from a table with a stage 2 predicate**
    - **Unless index on expression was used**
- **DB2 10 (CM)**
  - **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**
  - **Indexes can be utilized**
    - **A stage 2 predicate can be used for index screening**

DB2 11
Stage 2 predicate pushdown for list prefetch access

Residual Predicate Pushdown

- **Predicate pushdown applies to**
  - **Basic predicate (COL op value)**
  - **BETWEEN predicate**
  - **NULL predicate**
  - **Certain types of expressions**
    - **Built-in scalar functions**
      - **SUBSTR, UPPER, LOWER, DATE, MONTH…**
    - **Built-in operator functions**
      - **+, -, /, …**
    - **Constant, column name, host variable, special register, labeled duration**
    - **Cast-specification**
    - **Sequence object reference**
      - **PREVVAL only**
  - **Columns of all data types**
    - **Except LOB and XML**

Residual Predicate Pushdown – PUSHDOWN Column

```
EXPLAIN ALL SET QUERYNO = 1234 FOR
SELECT EMPID FROM EMPLOYEE
WHERE DEPT = 10
AND SUBSTR(PHONE, 1,3) = '217'
```

Whether the predicate is pushed down to the Index Manager or Data Manager for evaluation:

DSN_FILTER_TABLE

QUERYNO	PREDNO	STAGE	PUSHDOWN
1234	2	MATCHING	
1234	3	STAGE2	I

I - Index Manager evaluates the predicate
D - Data Manager evaluates the predicate
blank – no push down

Conversion of Stage 2 Predicates to Indexable – DB2 11

- **Prior to 11**
  - **Several common stage 2 predicates were still not indexable**
- **DB2 11 (CM after Rebind)**
  - **More stage 2 predicates are converted to be indexable**

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

- **If an qualifying index on expression exists, conversion will not occur**
- **Works with literals, host variables, parameter markers**
- **Queries are rewritten to be indexable**

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

Not-indexable

→

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

Indexable

Additional Indexable Predicates – DB2 11

- Prior to DB2 11
  - CASE expressions were processed at stage 2
  - OR/IN and OR COL IS NULL predicates stage 2

  } *Not-indexable*
- DB2 11 (CM after rebind)
  - CASE expressions are now indexable
    - For local/join predicates (when CASE expression 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 → WHERE COL1 = ? OR COL1 IS NULL

  - Multi-index access allowed for IN/OR

WHERE COL1 = ? OR COL2 IN (1,2) → WHERE COL1 = ? OR COL2 = 1 OR COL2 = 2

View and NTE Merging

- In DB2 9
  - DB2 performed materialization on some queries with views and nested table expressions
  - Merge
    - View or NTE syntax physically merged with referencing portion of statement
  - Materialization
    - View or NTE materialized into a work file then read by referencing portion of statement
- DB2 10 (CM)
  - Can merge views and NTEs in outer joins more often
    - Queries with IFNULL, NULLIF, COALESCE, CASE, VALUE expressions in the preserved row table of an outer join
    - When subquery predicate in table expression or single table view is on null supplying table
    - When query contains a correlated table expression without a GROUP BY, DISTINCT, or column function
    - Subquery in view or table expression is on preserved side of outer join

View and NTE Merging - Example

- Can merge views and table expressions in outer joins for
  - Queries with IFNULL, NULLIF, COALESCE, CASE, VALUE expressions in preserved row table of an outer join
    - Except when CASE expression merges to become a join predicate

```
SELECT A.COL1, B.COL1, A.COL2, B.COL2
FROM TAB1, (SELECT COALESCE(COL1, 0) AS
COL1 ,COL2
FROM TAB2 ) A
LEFT OUTER JOIN
(SELECT COALESCE(COL1, 0) AS COL1 ,COL2
FROM TAB3 ) B
ON A.COL2 = B.COL2
WHERE TAB1.COL2 = A.COL2;
```

Pre-DB2 10 this is materialized

DB2 10 will rewrite query to avoid materialization

```
SELECT COALESCE(A.COL1, 0) AS
COL1, B.COL1, A.COL2, B.COL2
FROM TAB1, TAB2 A
LEFT OUTER JOIN
(SELECT COALESCE(COL1, 0) AS COL1
,COL2 FROM TAB3 ) B
ON A.COL2 = B.COL2
WHERE TAB1.COL2 = A.COL2;
```


View and Table Expression Predicate Pushdown – DB2 11

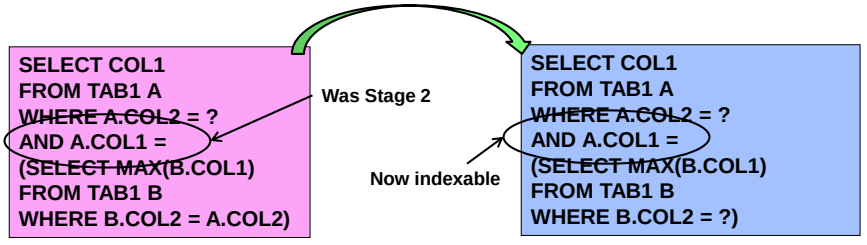
- Issue
  - Materialized view and table expressions can be expensive
  - Need to be able to push predicates into view or table expression
- DB2 11
  - Provides additional pushdown into views and table expression
    - Non-boolean term (OR) predicate
    - Stage 2 predicates (expressions)
    - Outer 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 Query Rewrite

- Prior to DB2 10
  - DB2 had limited ability rewrite correlated queries to be non-correlated
- 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



Correlated to Non-Correlated Query Rewrite – DB2 11

- 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 and UNION
- DB2 11
  - Correlated subqueries are converted to non-correlated subqueries
    - When covered by local predicate
    - Including non-boolean term predicates and UNION
  - Reusable workfile support for repeated access for correlated subqueries

Query Sort Enhancement

- Prior to DB2 10
  - Tournament tree sort is avoided
    - For small number of rows for FETCH FIRST N ROWS with ORDER BY
      - No need to sort entire answer set
      - Only top N are sorted
      - ORDER BY exploits FETCH FIRST n ROWS
        - Work files are not created (less I/O)
        - Only if (N * (sort key + data)) < 32 KB
  - Avoids allocating work files for small sorts for final sorts only
    - If no greater than 255 rows are to be sorted
      - And result is < 32 KB (sort key + data)

```
SELECT  ACCT_ID,  ACCT_STATUS
FROM    ACCOUNT
WHERE  ACCT_ID  IN
          (SELECT ACCT_ID
            FROM  BAD_ACCOUNTS
            ORDER BY AMOUNT_OWED DESC
            FETCH FIRST 5 ROWS ONLY)
```

Sorted in memory
*if (N * (sort key + data)) < 32 KB*

Query Sort Enhancement (cont..)

- DB2 10
  - Allows for larger results to avoid work file sorts
    - (N * (sort key + data)) < 128 KB
  - Sort avoidance for small sorts to intermediate sorts
    - Except for parallelism or SET functions

```
SELECT  ACCT_ID,  ACCT_STATUS
FROM    ACCOUNT
WHERE  ACCT_ID  IN
          (SELECT ACCT_ID
            FROM  BAD_ACCOUNTS
            ORDER BY AMOUNT_OWED DESC
            FETCH FIRST 5 ROWS ONLY)
```

Sorted in memory
*if (N * (sort key + data)) < 128 KB*

  - Hashing technique(new) for managing large sorts
    - Reduces number of merge passes needed to complete sorts
    - Can be used to identify duplicates on input to sort with functions
      - Such as DISTINCT or GROUP BY

Sort Performance – DB2 11

- In Memory Storage for Queries
  - Can specify the maximum allocation of storage for a query containing an ORDER BY clause, a GROUP BY clause, or both
  - Storage is allocated only during processing of the query
  - Increasing the value can improve performance of these queries
    - However, might require large amounts of real storage when several such queries run simultaneously
  - Range 1-128M, default=1M
    - Values – 1M to value specified in SORT POOL SIZE, whichever is larger
    - Used for final in-memory work file storage for final sort
      - Can use up to this value
    - Increase this value if more in-memory storage for final work file is needed

MAXSORT_IN_MEMORY

Enhanced Duplicate Removal – DB2 11

- Issue
  - Several queries require duplicate removal
    - DISTINCT, GROUP BY, etc.
  - Duplicate elimination via sorting can be expensive
- DB2 11
  - Provides new ways to removed duplicates
    - Index duplicate removal
      - IXSCAN_SKIP_DUPS (Y/N)
        - If duplicate index values were skipped during index scan
        - If sort was avoided

DSN_DETCOST_TABLE

    - Early out
      - EARLY_OUT (Y/N)
        - Whether fetching from the table stops after the first qualified row

Optimizer Statistics Validation/Index Probing

- Prior to 10
  - Poor access paths may be chosen when the optimizer has a bad filter factor estimate due to lack of accurate stats
- DB2 10 (CM) – always enabled…
  - DB2 uses RTS data and probes index non leaf pages to come up with better matching predicate filtering estimates
  - Estimate number of rids and/or validate number of table rows
    - Within start/stop key range by probing a limited number of index non-leaf pages
    - Validation of statistics from real-time statistics
- Runtime stats validation performed when
  - Query has matching index-access local predicate, and
  - Predicate contain literals (or REOPT(ALWAYS|ONCE|AUTO))
- At least one of the following conditions must also be true
  - Predicate is estimated to qualify no rows
  - Statistics indicate table contains no rows (or defaults exist)
  - Table is defined with VOLATILE
    - Or NPGTHRSH threshold is met

DSN_COLDIST_TABLE *contains estimates*

Optimize for 1 Row Enhancement – DB2 10 Post GA

- Prior to 10
  - OPTIMIZE FOR 1 ROW
    - Attempts to choose an access path that will avoid a sort
      - In order to return the 1st row quickly
    - Sort avoidance for this still allowed a cost based decision for a sort path to be chosen if estimated to be efficient
  - Enhancement prior to DB2 10 allowed the optimizer to block sort plans if OPTIMIZE FOR 1 ROW was coded
    - And at least 1 sort avoidance plan existed
- DB2 10 (post GA APAR PM56845)
  - Some OPTIMIZE FOR 1 ROW queries were switched from matching index access with sort, to a sort avoidance plan
    - But it was less efficient
  - Example: may have chosen a non-matching index scan to avoid sort instead of matching index plan that sorted
    - A non-matching index scan can be an inefficient choice
      - Because a large number of rows may need to be scanned to find 1st row qualifying row

Optimize for 1 Row (cont..)

- New DSNZPARM

OPT1ROWBLOCKSORT

- OPT1ROWBLOCKSORT
  - ENABLE
    - DB2 will disable sort access paths if a no-sort choice is available
  - DISABLE (default)
    - DB2 will strongly discourage sort access paths
    - And is unlikely to choose sort access paths
      - But there will be a chance a sort access path can win

OPTIOWGT

- OPTIOWGT
  - DISABLE (default in V8)
  - ENABLE (default in DB2 9/10) to stabilize access path selection
    - Adjusts downward weight of I/O cost in access path selection
    - Important when running on fast processors (z10, z196, ...)
  - Needed changed OPTIOWGT to ENABLE in DB2 10
- APAR PM70046
  - Sets OPTIOWGT=ENABLE

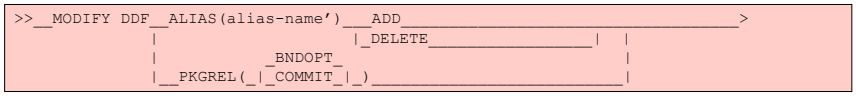
OPTIOWGT

RELEASE(DEALLOCATE) Support for Distributed Threads

- Prior to DB2 10
  - DRDA threads always allocated packages with RELEASE(COMMIT)
    - Allow DDL and BIND to interrupt thread
- DB2 10
  - RELEASE(DEALLOCATE) allowed for distributed packages
  - Reduce CPU consumption by avoiding
    - Repeated package allocation/deallocation
    - Processing going inactive/active
  - Provides higher CPU reduction for short transactions
    - For highly active threads (protected threads)
  - xPROCs, CTs and PTs, lookaside and prefetch not re-initialized
  - Benefits thread pooling and CMTSTAT=INACTIVE
  - Real memory savings and DBM1 virtual storage
    - Reduces number of distributed threads
  - Thread stays active if at least one DEALLOCATE package exists
    - Turns inactive after 200 times to free up distributed threads
    - Normal idle thread time-out detection applies

MODIFY DDF to Change

- -MODIFY DDF PKGREL command
  - Alters DDF's inactive connection processing
    - Applies only to CMTSTAT=INACTIVE
  - PKGREL(BNDOPT) honors package bind option
  - PKGREL(COMMIT) forces RELEASE(COMMIT)
    - Allow BIND and DDL concurrent with distributed work
  - PKGREL(DEALLOC) forces RELEASE(DEALLOCATE)
    - Provides better performance
    - BIND and DDL can be concurrent distributed work



-MODIFY DDF PKGREL (BNDOPT)

Recommended Usage (cont..)

- To minimize use of CPU resources for package allocation and deallocation
  - During production operating hours
    - Allocate/deallocate according to RELEASE(DEALLOCATE) bind option

MODIFY DDF PKGREL(BNDOPT)

  - During batch utilities and emergency maintenance
    - Use RELEASE(COMMIT)

MODIFY DDF PKGREL(COMMIT)

- Command results are not immediate
  - Access threads are terminated and made inactive at next commit
  - RELEASE(DEALLOCATE) threads remaining active waiting for a new UOW request from a client are terminated by a DDF service task(every 2 min)
  - When new UOW creates a database access thread
    - Packages are allocated using RELEASE(COMMIT)

RELEASE(DEALLOCATE) – DB2 11

- Issue prior to 11
  - RELEASE(DEALLOCATE)
    - Used often for performance reasons
    - However, in the case of persistent threads the thread might 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)

RELEASE(DEALLOCATE) – DB2 11

- DB2 11
  - PKGREL_COMMIT online system parameter
    - 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, utility needing to quiesce or invalidate application's package
      - DB2 implicitly deallocates/releases package at COMMIT or ROLLBACK
    - No longer 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 the packages
      - Behavior is same as if it was bound RELEASE(COMMIT)
- Not supported for the following
  - 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

PKGREL_COMMIT

RETURN TO CLIENT

- Prior to DB2 10
  - A stored procedure can only return result sets to immediate caller
  - If stored procedure is in a chain of nested calls
    - Result sets must be materialized at each intermediate nesting level
      - Often in a declared global temporary table
- DB2 10 - NFM
  - A result set can be returned from a stored procedure at any nesting level directly to the client calling application
    - No declared temporary table needed

DECLARE CURSOR…
WITH RETURN TO CLIENT

  - Result sets are not visible to any stored procedures at intermediate levels of nesting
    - Only visible to client that issued initial CALL
  - Not supported for stored procedures called directly or indirectly from triggers or functions

RETURN TO CLIENT - Performance

- Declared global temporary table usage is more expensive
  - Due to the catalog restructuring in DB2 10
  - May experience performance degradation when using declared temps
    - Due to materialization of result sets within stored procedures
- Using RETURN TO CLIENT
  - Can significantly improve performance
  - Especially for large result sets
    - No longer have a large temporary table passed through each nesting level

Currently Committed Data Access

- Issue
  - Applications acquire locks on data
    - Resulting in overhead, contention, concurrency issues
    - Possible timeouts
  - UR avoids contention
    - Can return uncommitted data
- DB2 10(NFM)
  - Allows access to version of data which was last committed
    - Version of data that existed before the blocking unit-of-work has changed the row
      - But has not yet committed the change
  - Ability to return currently committed data without waiting for locks
  - For uncommitted inserts or deletes
    - Not for uncommitted updates
  - Referenced data must be in a UTS

Binds and Prepares for Currently Committed Usage

- **BIND**

  CONCURRENTACCESSRESOLUTION
  (USECURRENTLYCOMMITTED | WAITFOROUTCOME)

- **PREPARE**

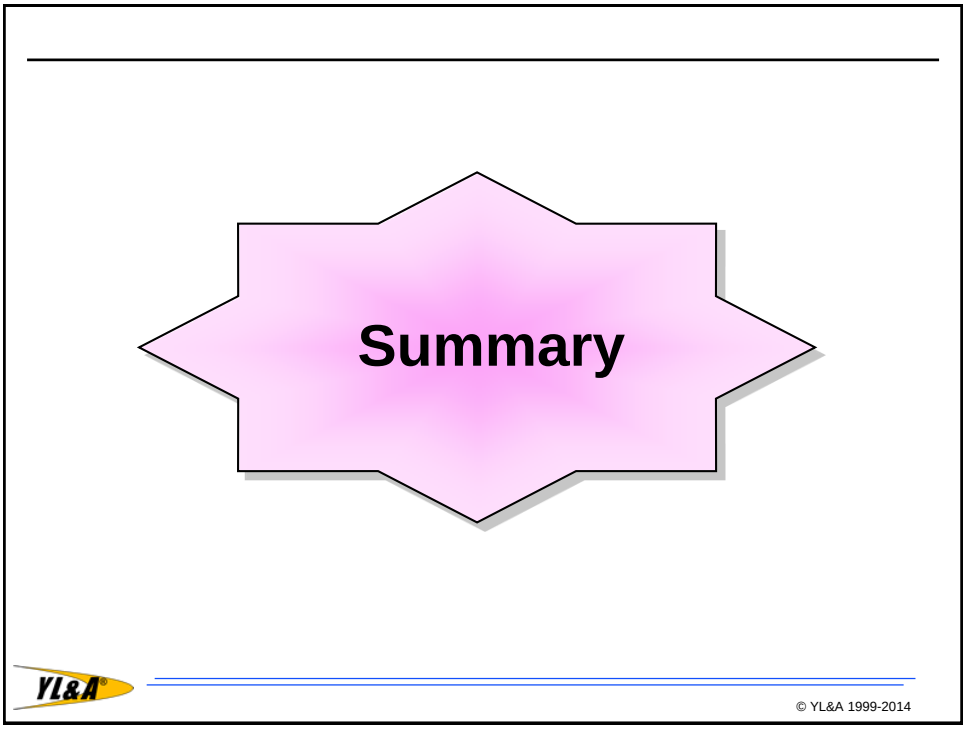
  USE CURRENTLY COMMITTED | WAIT FOR OUTCOME

- **CREATE/ALTER of PROCEDURE or FUNCTION**

  CONCURRENT ACCESS RESOLUTION
  USE CURRENTLY COMMITTED / WAIT FOR OUTCOME

Native SQL Procedure Performance

- **DB2 9**
  - Native SQL Procedures were introduced
  - Procedures execute in DBM1, not WLM address space
    - Performance improvements due to less cross-memory calls
- **DB2 10 (CM)**
  - Additional performance optimization
  - Specific CPU reduction in commonly used area
  - Section load avoidance with SET statements with function
  - Path length reduction in IF statement
  - Optimization in SELECT x from SYSDUMMY1
  - Chained SET statement support (NFM)
  - Potential OLTP workload performance improvements
    - CPU reduction
    - DBM1 Below the bar usage reduction
    - Response time improvement
  - Will need to drop/recreate existing procedures



Summary – DB2 10 and 11 Performance Opportunities

- DB2 10 and 11 for z/OS offers many opportunities for improving performance
  - Database Features
    - Member cluster on UTS
    - Index pseudo-delete cleanup
    - Inline LOBs
  - SQL and Application Features
    - Improved access paths
    - Deallocate for distributed threads
    - More indexable predicates
    - XML update/insert improvements
  - System Features
    - Virtual storage relief
    - In-memory tables
    - 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

YL&A

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 Certification Crammer
  - Covers exam 610 and 530
- DB2 10 for z/OS 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!!!!