Db2 11 and 12 for z/OS What Can I Take Advantage Of?

YLA Inc.

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.**

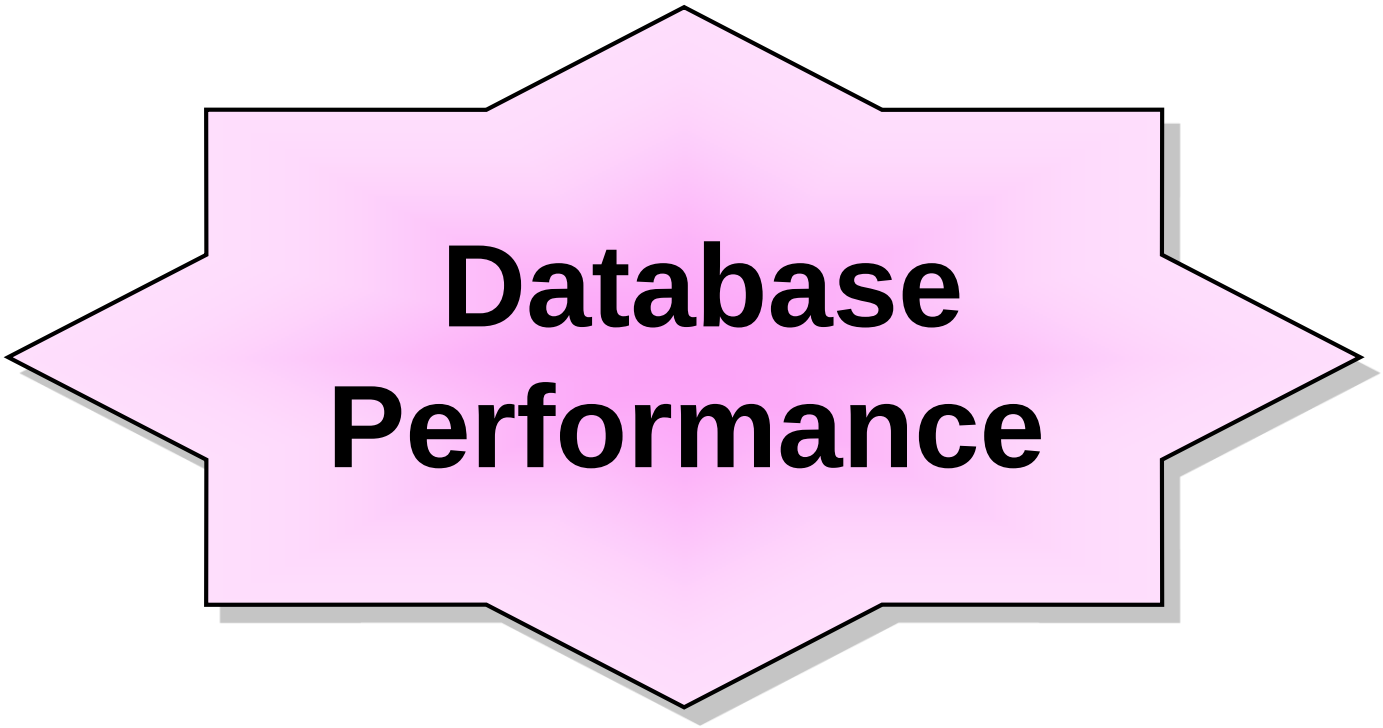
Abstract

With every new release of Db2 we look to see what features will allow us to improve the capabilities and performance of our existing applications as well as the availability of our data. We also have to plan to utilize new features in our development efforts. This presentation takes a look at the features in Db2 11 and those retrofit from 12, that will improve our performance and provide us with maximum data availability as well as advanced application development. We will be focusing on mostly on Db2 11 features and usage that can be utilized by DBAs and application programmers. Some Db2 12 performance features will be presented to introduce cool features that we can start to plan on utilizing in our databases and applications.

- **Overall improvements and migration goals in Db2 11 and 12**
- **Features for improved application and database performance**
- **Features for improved data availability**
- **Features for advanced application development**
- **Implementation and expectations of these new features**

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



Db2 11 - 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**

CREATE INDEX…
EXCLUDE NULL KEYS

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**

Account

	Acct_Num	Balance	Type	User_ID
x	~~1234~~	~~400.00~~	~~Check~~	~~1111~~
	3456	600.00	CD	5662
x	~~5672~~	~~800.00~~	~~Check~~	~~4579~~
	4567	200.00	Saving	2222

IK1 X	*RID1*
IK2	RID2
IK3 X	*RID3*
IK4	RID4
IK5	RID5
IK6	RID6

Acc_Num Index

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 REORGPSEUDODELETES 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

PSEUDO_DELETED_ENTRIES

Db2 11 - 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 (10)
    - Can also be controlled via catalog table

INDEX_CLEANUP_THREADS > 0

SYSINDEXCLEANUP

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

Db2 11 - 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**

Indirect Near	Refs Increase	Far	Increase	Row # Rows	Count Increase
0		0		6,606,093	0
38,622	38,622	82,048	82,048	6,967,418	361,325
38,638	16	185,339	103,291	6,977,109	9,691
38,645	7	243,914	58,575	6,988,602	11,493

Db2 11 - 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)

PCTFREE FOR UPDATE

PCTFREE_UPD

Db2 11 - 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**
    - **Larger number of partitions, larger 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)**

Db2 12 – PBR Table Space Size Limitations

- **Prior to Db2 12**
  - **Range partitioned table spaces were limited in size (128 TB)**
  - **This also was further limited depending on DSSIZE and page size**
    - **To alter DSSIZE entire table space needs REORG**
  - **DSSIZE is same for all parts**
  - **Could create design issues if there was need to growth beyond 256 parts with a 4K page, and 64GB DSSIZE….**

DSSIZE	4K	8K	16K	32K
4 GB	4,096	4,096	4,096	4,096
8 GB	2,048	4,096	4,096	4,096
16 GB	1,024	2,048	4,096	4,096
2 GB	512	1,024	2,048	4,096
4 GB	256	512	1,024	2,048

Db2 12 – Relative Page Number PBR Table Spaces

- Prior to Db2 12
  - An absolute page number was used
    - Internal page numbering is kept as a 4-byte value that includes a partition number and page number
  - Distinguishing which bits represent the partition and which represent the page number requires a shift value
    - LOG base 2 (DSSIZE/(page size))
  - Created a dependency on the page size and partition size which ultimately limited the number of partitions
- Db2 12
  - Introduces a number relative page number
  - Internal page numbering is kept as a value without a partition number
  - Page number is a relative page from the start of the partition
    - Partition number is kept only in the header page

UTS PBR RPN

Db2 12 – Range-Partition Table Space with RPN

- Db2 12
  - Range-partitioned(PBR) table spaces lifted many restrictions and improvements in space allocation
    - Data partition sizes can be up to 1TB
    - Greater flexibility in growing partitions
      - Can grow by any number of gigabytes
        - Instead of gigabytes in powers of 2
    - Table space can grow up to 4PB
    - No dependency between number of partitions and size of partition
    - DSSIZE can be increased for individual partitions as an immediate ALTER
      - Without requiring a REORG
      - A DSSIZE decrease would require REORG
    - Default established by zparm
    - Can also be established on CREATE/ALTER

PAGESET_PAGENUM

PAGENUM RELATIVE/ABSOLUTE
on CREATE/ALTER TABLESPACE

Db2 12 – Insert Partition with Specified Keylimit

- Prior to Db2 12
  - Partitions could only be added to the end of a table
  - There may be a need to add a partition in the middle of a table
    - Partitions may have outgrown their limits
  - Sometime we had designed with ‘free’ partitions in anticipation of this
    - Using alter limit key and reorg to rebalance across ‘new’ partition
- Db2 12
  - Can now insert a partition anywhere in the table
  - Allows for the insertion of a new partition with specified key value
    - Splits existing partition
    - Distribute data between the old and new partition
  - Online change
    - Deferred alter
    - Only affected parts need REORG
  - No PIT recovery allowed after REORG

ALTER TABLE ADD PARTITION ENDING AT

Db2 11 - 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 transitively 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

Db2 11 - 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

Db2 11 - Incremental Bind/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

Db2 11 - 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

Db2 11 - 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)
```

Db2 11 - 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)

Db2 12 – Basic Row Format Deprecation

- **Prior to Db2 12**
  - Use of reordered record format was determined by the RRF zparm
    - ENABLE (default) or DISABLE
  - Use could also be determined/changed with the REORG setting
    - ROWFORMAT (BRF/RRF)

BRF

NAME(C)	ADDRESS (V)		PHONE(C)	JOB_DESC (V)	
LAWSON	12	1234 YLA Way	217-555-1162	8	The Boss

← *VARCHAR placement as CREATED*

RRF

NAME(C)	PHONE (C)	OFF SET2	OFF SET4	JOB_DESC(V)	ADDRESS(V)
LAWSON	217-555-1162	18	20	The Boss	1234 YLA Way

← *VARCHARs moved to row end*

- **Basic row format is deprecated**
  - Basic row format page sets still supported
  - ZPARM and REORG options removed
  - RRF will only be used for new tables
    - Cannot be changed

Db2 12 – Fast Insert Algorithm

- **Fast insert algorithm**
  - **Can improve performance of INSERT operations for unclustered data**
  - **Can result in an increase in insert throughput**
    - **Especially with data that is not indexed**
  - **Can reduce logging and reduce class 2 elapsed time and class 2 CPU time**
  - **Only applies to UTSs with MEMBER CLUSTER**
    - **Is the default algorithm**
    - **Not applicable to other table space types**
- **Controlled via new ZPARM - DEFAULT_INSERT_ALGORITHM**

DEFAULT_INSERT_ALGORITHM

Db2 12 – Fast Insert Algorithm (cont..)

- **CREATE TABLESPACE or ALTER TABLESPACE statement**
  - **New option - INSERT ALGORITHM**

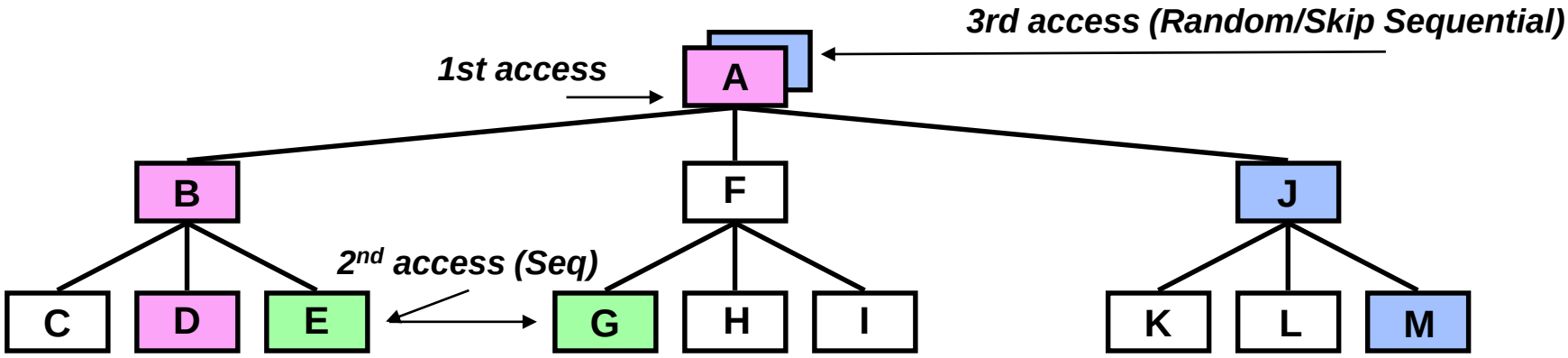
INSERT ALGORITHM

- **INSERT ALGORITHM**
  - **Specifies the algorithm that is used when rows are inserted into tables in this table space**
  - **The insert algorithm level is used only where applicable (MEMBER CLUSTER UTS)**
    - **0(default) - insert algorithm level is determined by the DEFAULT_INSERT_ALGORITHM zparm at the time a row is inserted**
    - **1 - basic insert algorithm is used**
    - **2 - fast insert algorithm is used**

DEFAULT_INSERT_ALGORITHM

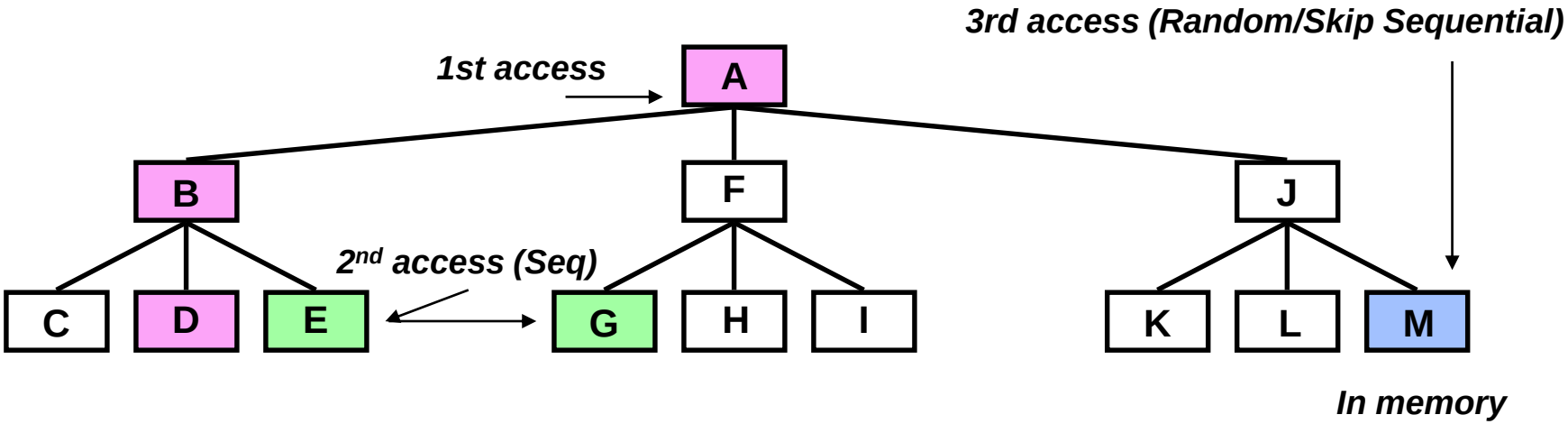
Db2 12 - In Memory Indexes

- **Prior to Db2 12**
  - **Index Manager maintains transversal information in index lookaside**
    - **Keeps track of pages visited when access required index leaf page**
      - **Kept from one access to next**
    - **Method is good for sequential access of an index**
      - **However for a random access of the index, several getpages are required if the look aside does not get a parent of the leaf**



Db2 12 - In Memory Indexes

- **Db2 12**
  - **Introduces index Fast Traverse Block (FTB) with the Index Manager**
    - **Optimizes memory structure for fast index lookups and improves random index access**
  - **Complement to index lookaside**
  - **Best for**
    - **Indexes supporting heavy read access**
    - **indexes on tables with a random insert or delete pattern**
    - **Indexes with high PCTFREE**



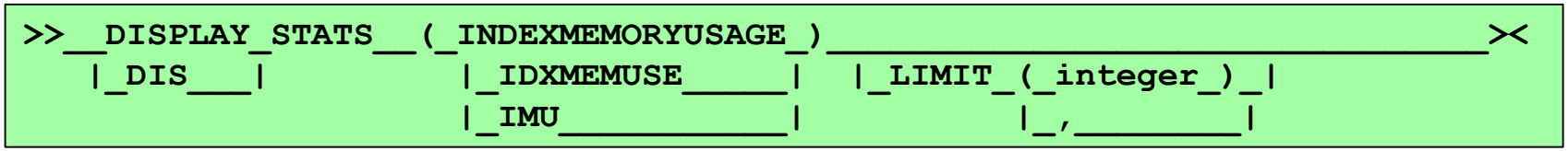
Db2 12 - In Memory Indexes

- Db2 12
  - Fast index lookups
    - Optimized memory structure
      - New index Fast Traverse Block (FTB)
    - In memory area outside buffer pool
    - Only for indexes defined as unique (<64 byte key size)
  - Benefits
    - CPU reduction for index look ups
  - Usage determined by Db2

INDEX_MEMORY_CONTROL

  - DSNZPARM that controls the amount of memory used for FTB

AUTO – upper limit set at 20% of allocate buffer pool
DISABLE – no storage used for FTB
50-200000 MB – storage to be used



Db2 12- Non-Deterministic Expression for Column Auditing

PM99683 and PI15298

Which users and what operation changed the data?

```
CREATE TABLE ACCOUNT
(ACCT_ID CHAR(4) NOT NULL ,
 BALANCE INT NOT NULL,
 TYPE CHAR(2) NOT NULL ,
 SQLID VARCHAR(8) GENERATED ALWAYS AS (CURRENT SQLID),
 DCOP CHAR(1) GENERATED ALWAYS AS (DATA CHANGE OPERATION),
 SYS_START TIMESTAMP(12) NOT NULL GENERATED ALWAYS
 AS ROW BEGIN,
 SYS_END TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END,
 TRANS_ID TIMESTAMP(12) GENERATED ALWAYS
 AS TRANSACTION START ID ,
 PERIOD SYSTEM_TIME(SYS_START, SYS_END));

CREATE TABLE ACCOUNT_HIST
(ACCT_ID CHAR(4) NOT NULL ,
 BALANCE INT NOT NULL,
 TYPE CHAR(2) NOT NULL ,
 SQLID VARCHAR(8) ,
 DCOP CHAR(1) ,
 SYS_START TIMESTAMP(12) NOT NULL,
 SYS_END TIMESTAMP(12) NOT NULL,
 TRANS_ID TIMESTAMP(12));

ALTER TABLE ACCOUNT ADD VERSIONING USE HISTORY TABLE ACCOUNT_HIST
ON DELETE ADD EXTRA ROW ;
```

CURRENT SQLID
SESSION_USER (USER)
CURRENT SERVER
etc..

Insert, Update, Delete

CURRENT SQLID
SESSION_USER (USER)
CURRENT SERVER
etc..

Insert, Update, Delete

Db2 12 - Non-Deterministic Expression for Column Auditing

- Prior to 12
  - System-temporal tables track changes in data
  - Columns ROW BEGIN and ROW END are used and populated by Db2 based on system clock
    - Only tracks time change is made to base table data
- Db2 12 (Db2 11 with APARs PM99683 and PI15298)
  - Additional columns are added to temporal tables
    - Tracks who, how, and what when changes are made to the base table
  - Columns are automatically maintained by Db2
    - Avoids need to write complex and often error-prone logic in user applications to populate columns (often done via triggers)
  - Performance benefit
    - Performance of populating auditing columns by using this Db2 feature
      - Compared to populating non-generated columns using triggers
      - Use of built-in Db2 support for maintaining auditing columns uses
        - 14% less class 2 CPU (IBM measurement - compared to using user-defined triggers)
    - Can save on CPU processing time by using non-deterministic expressions for column auditing
      - Reduce the complexity and overhead in application

Subsystem Performance

Db2 11 - 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

VPSIZE / total pages read/second(including pages prefetched)

VPSIZE * (1-VPSEQT/100)/random synch I/O/sec

If this is larger - sequential getpages and prefetch are affecting residency time

Db2 11 - Random vs. Sequential Getpages

Prior to Db2 11

- Getpages - sequential
  - Table scan sequential prefetch
  - LOB access
- Getpages - random
  - Random read
  - Random no read (seq inserts)
  - Dynamic prefetch
  - List prefetch
  - Sequential format writes (no read)

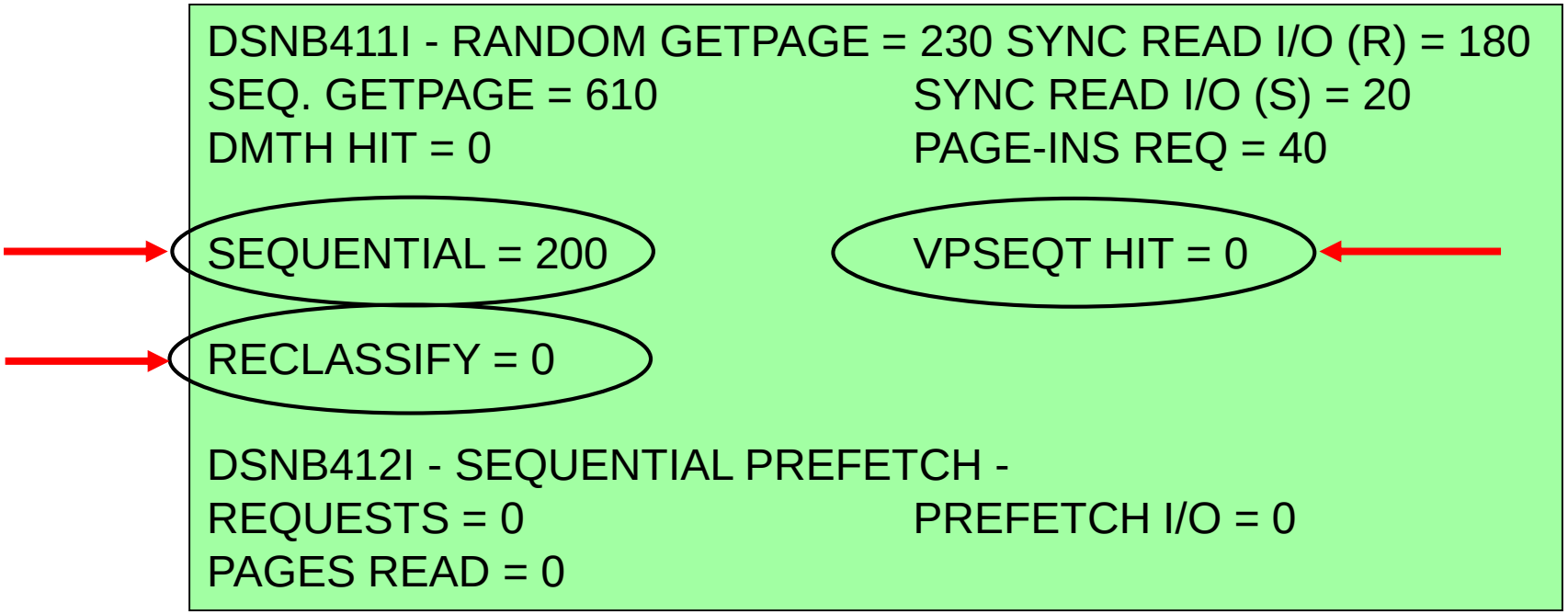
Db2 11 - more accurate random buffer hit ratio and number of truly random synchronous read I/Os

Sequential inserts that do no reads remain as most significant exception where buffers are incorrectly classified as random

Db2 11

- Getpages - sequential
  - Table scan sequential prefetch
  - LOB access
  - Dynamic prefetch
  - List prefetch
  - Sequential format writes (no read)
- Getpages - random
  - Random read
  - Random no read (seq inserts)

Db2 11 - Page Classifications – Display Bufferpool Detail



Db2 10 - PM70981

- **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**

Primarily for IBM Service

Db2 12 (11 – PI22091) – Buffer Pool Size Simulation

- Issues
  - When allocating buffer pools the theory is the bigger the better
  - However, large amounts of storage may not be available, or maybe the larger buffer pools may not benefit the application
- Db2 12 (Db2 11 – PI22091)
  - Provide new function for users to measure the benefit of increasing the buffer pool size
  - Aids users in determining the benefit of increasing buffer pool sizes, support for buffer pool simulation
  - ALTER BUFFERPOOL command includes new SPSIZE and SPSEQT
  - With a simulated buffer pool, pages that are stolen from the virtual buffer pool are tracked in the simulated buffer pool
    - Db2 determines whether a read I/O could have been avoided if the virtual buffer pool had the additional buffers
  - The number of read I/Os that could have been avoided is displayed in DISPLAY BUFFERPOOL command is issued
  - New section for simulated buffer pool statistics in the IFCID 2 statistic

Db2 12 (11 – PI22091) – Buffer Pool Size Simulation

```
DSNB401I – BUFFERPOOL NAME BP0, BUFFERPOOL ID 0 USE COUNT 10
DSNB402I – BUFFERPOOL SIZE = 3000 BUFFERS AUTOSIZE – YES
                    VPSIZE MINIMUM   = 2350        VPSIZE MAXIMUM = 3125
                    ALLOCATED        =  3000      TO BE DELETED = 0
                    IN-USE/UPDATED   = 200
                    BUFFERS ACTIVE   = 3000
                    OVERFLOW ALLOC = 0
DSNB431I – SIMULATED BUFFER POOL SIZE = 0 BUFFERS –
                    ALLOCATED        = 0
                    IN-USE           = 0       HIGH IN-USER          = 0
                    SEQ-IN-USE       = 0       HIGH SEQ-IN-USE       = 0
```

```
DSNB461 – SIMULATED BUFFER POOL ACTIVITY -
                         AVOIDABLE READ I/O –
                         SYNC READ I/O (R)          =0
                         SYNC READ I/O (S)          =0
                         AYSNC READ I/O             =0
                         SYNC GBP READS (R)         =0
                         SYNC GBP READS (S)         =0
                         ASYNC GBP READS            =0
                  PAGES MOVED INTO SIMLUATED BUFFER POOL = 0
                  TOTAL AVOIDABLE SYNC I/O DELAY = 0 MILLISECONDS
```

Db2 12 – Overflow Area for PGSTEAL(NONE) Buffer Pools

- **Issue**
  - **PGSTEAL (NONE) is a buffer pool page stealing option that allows for pages to remain in the buffer pool with the overhead of maintaining a queue (LRU queuing) or prefetch scheduling**
- **Db2 12**
  - **With PGSTEAL(NONE) no page stealing occurs if buffer pool is large enough to contain all assigned open objects**
  - **Db2 pre-loads the buffer pool when object is opened**
  - **Db2 implicitly creates an overflow area for pages that do not fit in buffer pool**
  - **Overflow area is create when buffer pool is allocated**
  - **Size is based on VPSIZE (generally 10% of VPSIZE)**
  - **Page stealing may occur in overflow area**
- **DISPLAY BUFFERPOOL - OVERFLOW ALLOC**
  - **Number of allocated buffers in overflow area for a buffer pool using PGSTEAL(NONE)**

Db2 12 - Overflow

```
DSNB401I – BUFFERPOOL NAME BP0, BUFFERPOOL ID 0 USE COUNT 10
DSNB402I – BUFFERPOOL SIZE = 3000 BUFFERS AUTOSIZE – YES
                    VPSIZE MINIMUM   = 2350        VPSIZE MAXIMUM = 3125
                    ALLOCATED        =  3000       TO BE DELETED = 0
                    IN-USE/UPDATED   = 200
                    BUFFERS ACTIVE   = 3000
                    OVERFLOW ALLOC = 0
DSNB431I – SIMULATED BUFFER POOL SIZE = 0 BUFFERS –
                    ALLOCATED      = 0
                    IN-USE         = 0      HIGH IN-USER          = 0
                    SEQ-IN-USE     = 0      HIGH SEQ-IN-USE       = 0
```

```
DSNB416I – OVERFLOW RANDOM GETPAGE      =0
           OVERFLOW SYNC READ I/O (R)   =0
           OVERFLOW SEQ. GETPAGE        =0
           OVERFLOW SYNC READ I/O (S)   =0
```

SQL and Optimization

Db2 11 - 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

"Filter Factor Hints"

Db2 11 - 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

PLAN_TABLE

HINT_USED = APREUSE

- 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

APREUSE(WARN)

Db2 12 – Improvements to Plan Stability

- Db2 12
  - FREE PACKAGE
    - Can choose to FREE either ORIGINAL or PREVIOUS
    - Option to only FREE the package if it is invalid

      INVALIDONLY
    - Ability to FREE package copies without application outage
  - REBIND PACKAGE

    APREUSESOURCE
    - Option during SWITCH
    - Can choose ORGINAL or PREVIOUS for APREUSE
    - Ensure that SWITCH does not choose an invalid copy
    - No longer need to perform REBIND SWITCH with subsequent REBIND APREUSE

Db2 12 – EXPLAIN Table Creation

- New ADMIN_EXPLAIN_MAINT stored procedure
  - Create EXPLAIN tables
  - Upgrade them to the format for the current Db2 release
  - Complete other maintenance tasks

ADMIN_EXPLAIN_MAINT

- *mode* - Processing mode
  - RUN - Alter and create EXPLAIN tables for the specified SCHEMA
  - PREVIEW - No changes are processed
- *Action* - Action that is completed for EXPLAIN tables in specified schema
  - STANDARDIZE
    - Upgrade all existing EXPLAIN tables to current version
  - STANDARDIZE_AND_CREATE
    - Upgrade EXPLAIN tables to current version, and create any missing tables
  - CREATE
    - Create a new set of EXPLAIN tables in the specified schema
  - CREATE_ALIAS
    - Create a new set of aliases only
  - DROP
    - Drop all existing EXPLAIN tables
  - DROP_AND_CREATE
    - Drop all EXPLAIN tables and create a replacement

Db2 11 - 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

Db2 11- 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

Db2 12 – Automatic Profile Update with Recommendations

- **Prior to Db2 12**
  - **Statistics feedback was introduced in Db2 11**
  - **Feedback occurs at BIND, REBIND, and PREPARE**
    - **Externalized to SYSSTATFEEDBACK**
  - **During Explain**
    - **Externalized to DSN_STAT_FEEDBACK**
  - **The only option available was a manual update to RUNSTATS in order to collect the recommended statistics**
- **Db2 12**
  - **Can automatically apply statistics recommendations generated during query optimization to statistics profiles**
  - **Established with the new STATFDBK_PROFILE zparm**

STATFDBK_PROFILE

Db2 11 - RUNSTATS RESET

- **Issue:**
  - **Changes to target objects and many RUNSTATS invocations, with different options, might result in 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**
    - **Previous values cannot be recovered**
    - **Dynamic statement cache invalidated**
  - **To reset access path statistics with RUNSTATS**
    - **Specify RESET ACCESSPATH**

RESET ACCESSPATH

Db2 11- Stage 2 Predicates Pushdown – List Prefetch

- **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)**

DSN_FILTER_TABLE

QUERYNO	PREDNO	STAGE	PUSHDOWN
1234	2	MATCHING	
1234	3	STAGE2	I

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

Db2 11 - 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
```

Not-indexable

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

Indexable

Db2 11 - Additional Indexable Predicates

- 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 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 → 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
```

Db2 11 - View and Table Expression Predicate Pushdown

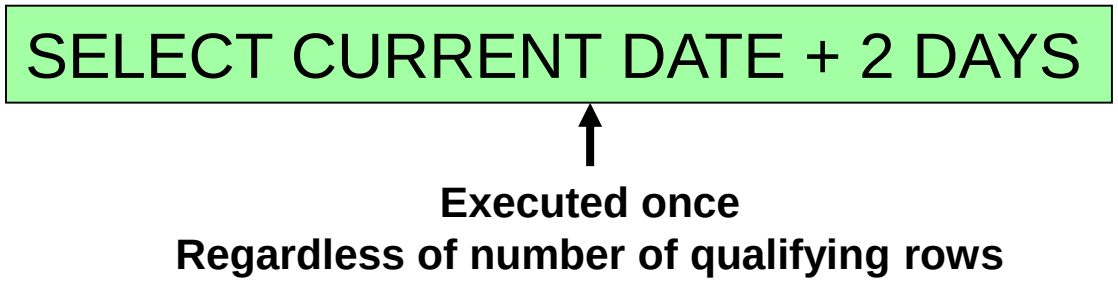
- **Issue**
  - Materialized view and table expressions can be expensive
  - Need to be able to push predicates into view or table expression
- **Db2 11**
  - Provides additional pushdown into views and table expression
    - Non-boolean term (OR) predicate
    - Stage 2 predicates (expressions)
    - 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

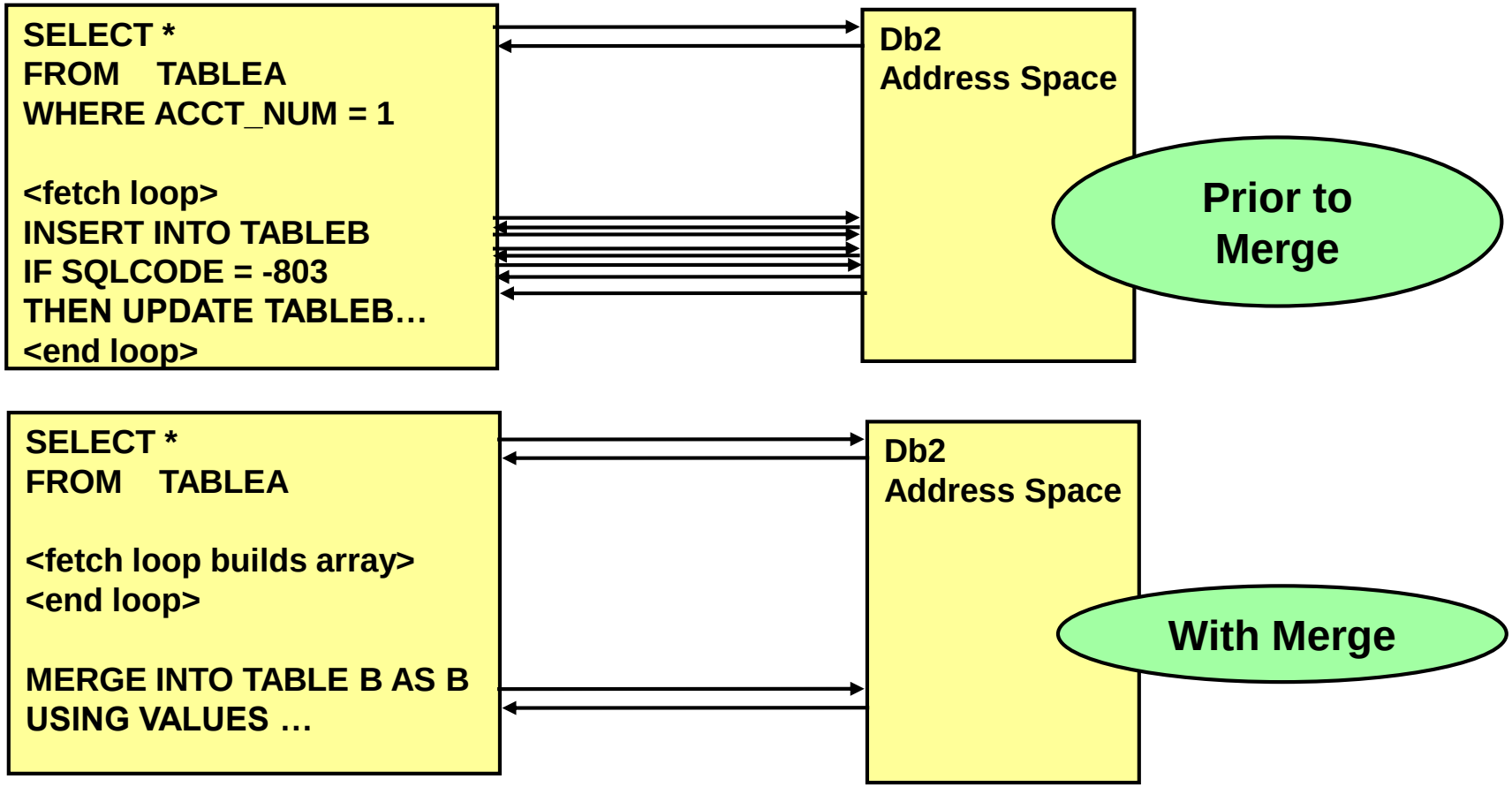
Db2 11 - Optimization of Invariant Expressions

- **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**



MERGE for Data Replication

- Using MERGE can greatly help reduce number of SQL statements when replicated/synchronizing data between database
- However, it was limited prior to Db2 12 as it did not allow for DELETE support and only supported UPDATE and INSERT



Db2 12 – Merge Enhancements

- Prior to Db2 12
  - The MERGE statement was incomplete as it did not allow for DELETE support
    - Only supported UPDATE and INSERT
  - It also did not have support for multiple MATCHED clauses
- Db2 12
  - Enhancements to the MERGE statement include support for
    - Table-reference as an alternative way of specifying source data for the MERGE statement
    - Multiple MATCHED clauses
    - Additional predicates with MATCHED or NOT MATCHED
    - DELETE operations
    - IGNORE and SIGNAL statement as action
  - More compatible now with other DBMS

Db2 12 – Fast Insert Algorithm

- **Fast insert algorithm**
  - **Can improve performance of INSERT operations for unclustered data**
  - **Can result in an increase in insert throughput**
    - **Especially with data that is not indexed**
  - **Can reduce logging and reduce class 2 elapsed time and class 2 CPU time**
  - **Only applies to UTSs with MEMBER CLUSTER**
    - **Is the default algorithm**
    - **Not applicable to other table space types**
- **Controlled via new ZPARM**

DEFAULT_INSERT_ALGORITHM

- **CREATE TABLESPACE or ALTER TABLESPACE statement**
  - **New option**

INSERT ALGORITHM

Db2 12 – Dynamic SQL Plan Stability

- **Prior to Db2 12**
  - **Unstable performance of repeating dynamic SQL statements**
  - **Environmental changes can result in change in access path or performance regression**
    - **Problematic to manage**
  - **Potential issues resulting from**
    - **Maintenance**
    - **Release migration**
    - **System parameter changes**
    - **Schema changes**
    - **Statistics (RUNSTATS)**

CACHEDYN_STABILIZATION

- **Db2 12**
  - **Introduces ability for dynamic SQL to have same performance advantages as dynamic**
  - **Db2 avoids processing full prepares when it uses the captured run time structures**

Db2 12 – Advanced Triggers

- **Prior to Db2 12**
  - **Db2 only had support for simple triggers**

- **Db2 12**
  - **Enhances trigger support**
  - **Introduces the concept of 'advanced triggers'**
  - **Simple triggers are now referred to as 'basic triggers'**
  - **Advanced triggers can:**
    - **Define and reference an SQL variable**
    - **Include more traditional SQL statements**
    - **Include SQL PL control statements**
    - **Include dynamic SQL statements**
    - **Reference a global variable, or assign a value to a global variable**
    - **Explicitly specify options, including bind options**
    - **Include SQL comments**
    - **Define multiple versions**

Db2 11 - Piece Wise Data Deletion

- Prior to Db2 12
  - A DELETE could potentially qualify a large amount of data
  - No way to perform interim COMMITs
  - Could lead to issues with locking and logging

```
DELETE FROM TABLE_ONE WHERE COL_ONE >10
```

- Db2 12
  - Provides for a piece wise data deletion
  - Goal is to lessen impacts of locking and logging
    - When millions of rows could be affected by a single statement

```
DELETE FROM TABLE_ONE WHERE COL_ONE > 10
FETCH FIRST 3 ROWS ONLY
```

  - Allow fullselect as the target of a delete statements
    - Where fullselect includes the FETCH FIRST n ROWS ONLY
  - Provides a way to perform interim commits

Db2 11 - RELEASE(DEALLOCATE) Thread Interruption

PKGREL_COMMIT

- Db2 11
  - PKGREL_COMMIT online DSNZPARM
    - 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

Db2 11 - 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**

ALTER TABLE TB1 PARTITION 1 ENDING AT ('0010')

REORP

- **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**

AREOR

ALTER TABLE TB1
PARTITION 1 ENDING AT ('0010')

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

Db2 11 - 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**

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

ALTER TABLE…
DROP COLUMN col1
RESTRICT

Cannot be dropped if any views, indexes, unique constraints, or referential constraints are dependent

Db2 11/12 PIT Recovery for Deferred Schema Changes

- Db2 11
  - Restriction lifted for some pending alters
  - PIT recoveries are now possible after a materializing REORG
    - For PBR, LOB or XML tablespaces
- Db2 12
  - Extends PIT recovery before materialization of pending definition changes
    - Column alterations can be pending definition changes
    - Can be a PBG table space
    - Can be ALTER TABLE DROP COLUMN
  - Additional pending column alterations
    - ALTER TABLE ALTER COLUMN statements that change the data type, length, precision, or scale of columns can now be pending alterations
    - By default, column alteration operations are immediate operations
      - Column alterations operation zparm
      - Set DDL_MATERIALIZATION = ALWAYS_PENDING

DDL_MATERIALIZATION

Db2 11/12 - Physical Deletion of Empty PBG Partitions

- **Prior to Db2 11**
  - **REORG of a PBG table space can result in empty physical partitions at end**
- **Db2 11**
  - **An empty trailing partition occurs when REORG utility moves all data records from a partition into lower numbered partitions**
  - **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**

REORG_DROP_PBG_PARTS

  - **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**
- **Db2 12**
  - **Ability to drop UTS PBG partitions**
    - **DROP_PART option in REORG**
    - **Remove empty UTS PBG part pruning when zparm not used**

Db2 11 - REORG without Sorting

- Prior to 11
  - Re-clustering during a REORG cannot be avoided
    - 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 RECLUSTER YES/NO option on SORTDATA NO
    - 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

Db2 12 – Point-In-Time Recovery Improvement

- **Prior to Db2 12**
  - **A RECOVER TOLOGPOINT or TORBA would process all objects even if they had not changed since the recovery point**
- **Db2 12**
  - **RECOVER utility using TOLOGPOINT or TORBA skips unnecessary recoveries by default**
  - **If an object has not been changed since the recovery point**
    - **Object is not recovered**
  - **Avoids unnecessary work and use fewer resources**
    - **Can also potentially reduce time that objects are unavailable and improve overall recovery time**
  - **Controlled by new SCOPE option**
    - **SCOPE UPDATE (default)**
      - **Unchanged objects are to be skipped during recovery**
    - **SCOPE ALL**
      - **All objects are to be processed by RECOVER**
        - **Even if they have not changed since the recovery point**
  - **Does not apply to RECOVER BACKOUT YES or RECOVER LOGONLY**

SCOPE UPDATE

Summary – Db2 11 and 12…Performance/Availability

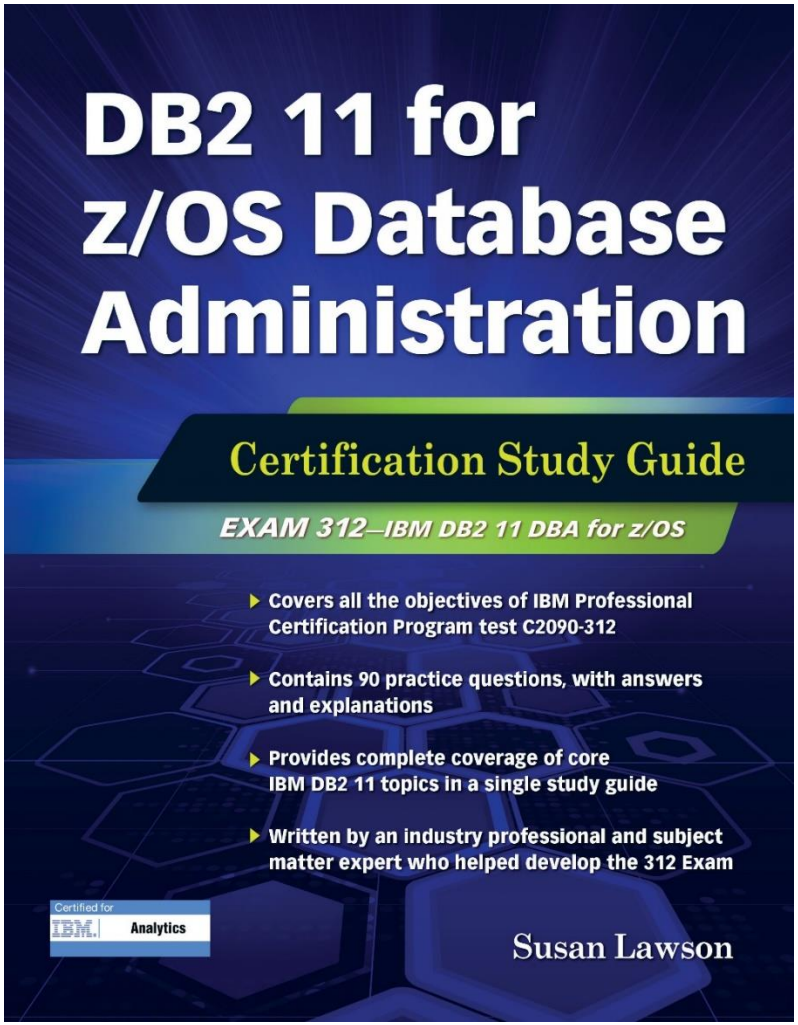
- Db2 11 and 12 for z/OS offers many opportunities for improving performance and increasing availability
  - Database Features
    - No log declared temps
    - Index pseudo-delete cleanup
    - Partition limit increase
  - SQL and Application Features
    - Improved access paths
    - Deallocate for 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

Db2 z/OS Database Administration – Certification Guide

Db2 11 for z/OS Database Administration Certification Study Guide

July 15, 2016

Authored by:
Susan Lawson



Db2 z/OS Courses and Performance Audits/Health Checks by YLA

- **Db2 z/OS Courses**
  - Db2 12 or 11 for z/OS Transition
    - Application or DBA or both
  - Db2 11 for z/OS DBA Certification Crammer
    - Covers exam 610 and 530
  - Db2 for z/OS High Performance Design and Tuning
    - Application, Database, Systems
  - Db2 for z/OS Data Sharing
    - Implementation, Performance, Recovery
  - Db2 for z/OS Application Development and SQL
    - Design, Tuning and Performance

All classes are customized based upon customer requirements

info@ylassoc.com

- **Db2 z/OS Performance and Availability Audits**
  - Existing or new database designs and applications
  - Health check of all the performance 'points' in a Db2 z/OS environment
    - Physical Table/Index Designs
    - Db2 Subsystem/Data Sharing Components
    - Application Code and SQL
  - Review designs/system for continuous availability opportunities
- ***Results***: problems identified, solutions proposed/implemented, continual knowledge transfer

Cost Avoidance through performance tuning!

YL&A®