Db2 12+/13 for z/OS Database Design and Application Performance: Features and Usage

Susan Lawson, YLA

Susan_Lawson@ylassoc.com

YLA Inc.

www.ylassoc.com
info@ylassoc.com

Susan Lawson – YL&A

Susan Lawson is an internationally recognized consultant and lecturer with a background in Db2 z/OS system and database administration for 34 years. She works with several large clients to help design, develop, implement, and tune some of the largest and most complex Db2 z/OS databases and applications. She also performs performance/availability audits and health checks for many clients to help reduce costs through proper performance tuning and to help ensure availability, stability and scalability. Her other activities include authoring books, articles, and white papers, and developing/teaching a variety of advanced database and application courses. She is an IBM GOLD Consultant and an IBM Champion.

www.linkedin.com/in/slawsonyla/

IBM Gold Consultant Excellence 2023

IBM

Awarded through the program

IBM Gold Consultant

IBM Champion 15 Year Milestone

IBM

Awarded through the program

IBM Champion

YL&A Services and Education

- Db2 z/OS Application Stabilization and Optimization
- Db2 z/OS Performance and Availability Audits
- Db2 z/OS Security Management Audits
- Db2 z/OS Systems Configuration and Management
- Db2 z/OS Version Migration and Maintenance
- Db2 z/OS Database Design and Administration
- Db2 z/OS Application Design and Development
- Db2 z/OS Performance Data Collection and Exploitation

www.ylassoc.com
info@ylassoc.com

Abstract

Db2 12+/13 for z/OS Database Design and Application Performance: Features and Usage

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 12 (+FLs) and 13, that will improve our performance and provide us with maximum data availability as well as advanced application development. We will be focusing on features (including those in the most current function level) that can be utilized by DBAs and application programmers.

- Overall improvements and goals in Db2 z/OS 12 (+FLs and APARs) and 13
- Features for improved application and database performance
- Features for improved application and database design
- Usage features for improved application performance
- Discuss new features delivered in latest function levels

Db2 12 and 13 Function Levels and APARs

- Since 2016 and Db2 M500 there have been 10 function levels and several APARs
  - Several of these delivered new or enhanced functionality
- Db2 13 for z/OS was delivered in May 2022 and FL 502 in October
  - Delivers many performance, usability, and scalability
- This presentation will look at those features related to application and database design, performance and usage

2016	Db2 12
2017	FL 501
2018	FL 502, 503
2019	FL 504, 505, 506
2020	FL 507, 508
2021	FL 509, 510
2022	Db2 13, FL 502
2023	FL 503, 504(Oct)

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 design changes
    - Plan for efforts needed
    - Evaluate effectiveness

Database Performance and Availability – Db2 12+/13

- Relative Page Numbering Default
- Improved Insert for RPN in Data Sharing
- PBG to PBR Conversion
- Remove Stack Limits for Pending Alters
- Max Parts Default Change
- Increased Number of Open Data Sets
- Improved Data Capture
- Larger Column Names
- Non-Deterministic Expression Column Auditing
- Segmented Multi-Table Tablespace to UTS

- UTS PBR Hidden ROWID
- Faster PBG Insert
- Improved Index-Lookaside
- FTB Support for Non-Unique Indexes
- FTB Selective Index Usage
- FTB INDEXTRAVERSECOUNT
- Online Load Replace
- Trigger, Temporal and Archive Transparency
- RTS Scalability Improvements
- LOB Compression Improvements
- Transparent Data Encryption Functions
- Huffman Compression

Application and SQL Enhancements – Db2 12+/13

- Profile Table Enhancements
- Controlling RELEASE via Profile Tables
- Improved Index Split Information
- Phased-In REBIND
- Statistics Deletion via Profiles
- SQL Data Insights
- Application Lock Timeout Control
- Deadlock Priority Control
- Global Variables for Locking Limits
- Lock Avoidance for Singleton Select

- Lock Avoidance for Singleton Select
- BTB Local Thread Storage Reduction
- Statistics Collection Improvements
- Reduced Sort Row Length
- Db2 SORTL Improvements
- Sort Improvement with Long Varchars
- List Prefetch for Merge
- LISTAGG and SUBSTR Performance
- HASH Function Usage
- Create/Replace for Procedures

Db2 z/OS 12 Database Design Performance Features and Usage

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
      - Includes a partition number and page number
- Db2 12
  - Introduces a 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
  - Absolute was still the default for creation
- Db2 13 (M100)
  - PAGESET_PAGENUM zparm default changes to RELATIVE
  - Partition-By-Range tablespace will be created by default
  - PBR RPN requires EA/EF datasets

PAGESET_PAGENUM

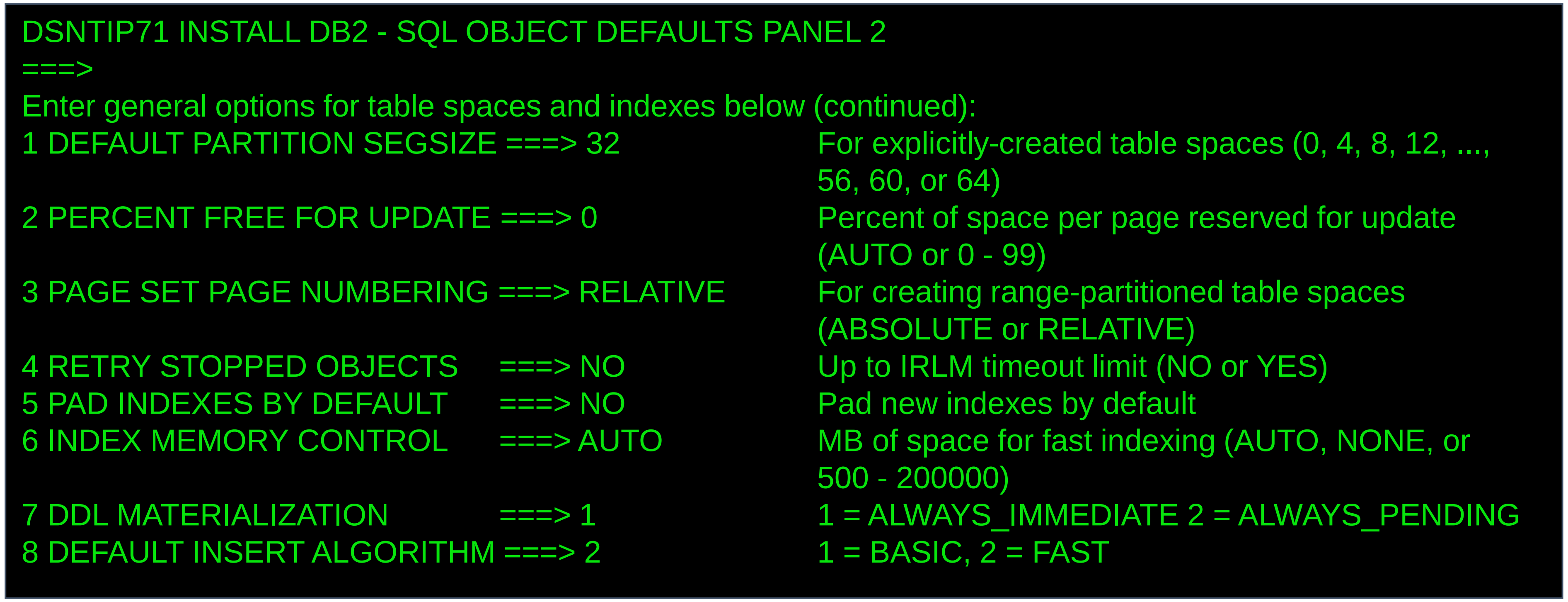
PBR Tablespace Creation – RPN Default

Å 5.н мо ớ а мллύ

Å t!D9{9¢ψt!D9b¦a ÎLJI·ǸƳ ŘŜŦI·ǳű ŎĶI·ƴƎŜă ű ƻ w9[!¢L±9

Å /ǈŜI·űλƻƴ ƻŦ t.w űI·ōſŜăLJI·ŎŜ ǿλűK ōŜ ŎǈŜI·űŜŘ I·ă wtb ōè ŘŜŦI·ǳfű

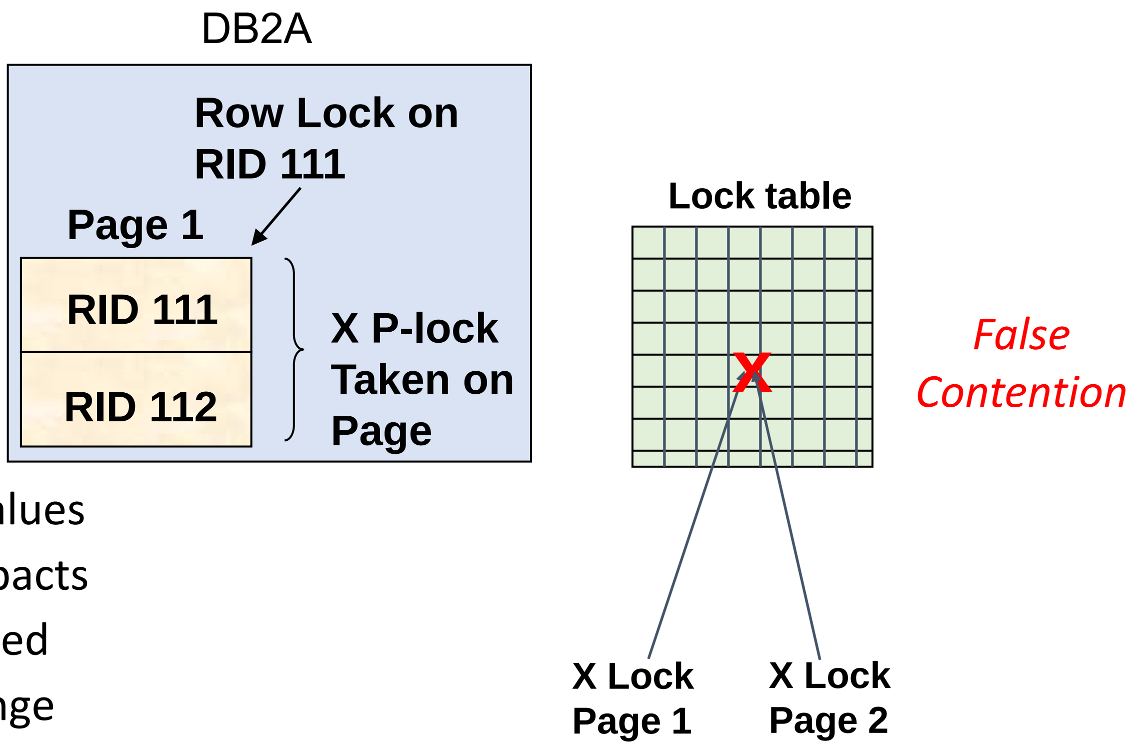
PAGESET_PAGENUM



```
DSNTIP71 INSTALL DB2 - SQL OBJECT DEFAULTS PANEL 2
===>
Enter general options for table spaces and indexes below (continued):
1 DEFAULT PARTITION SEGSIZE ===> 32          For explicitly-created table spaces (0, 4, 8, 12, ...,
                                             56, 60, or 64)
2 PERCENT FREE FOR UPDATE ===> 0             Percent of space per page reserved for update
                                             (AUTO or 0 - 99)
3 PAGE SET PAGE NUMBERING ===> RELATIVE      For creating range-partitioned table spaces
                                             (ABSOLUTE or RELATIVE)
4 RETRY STOPPED OBJECTS    ===> NO           Up to IRLM timeout limit (NO or YES)
5 PAD INDEXES BY DEFAULT   ===> NO           Pad new indexes by default
6 INDEX MEMORY CONTROL     ===> AUTO         MB of space for fast indexing (AUTO, NONE, or
                                             500 - 200000)
7 DDL MATERIALIZATION      ===> 1            1 = ALWAYS_IMMEDIATE 2 = ALWAYS_PENDING
8 DEFAULT INSERT ALGORITHM ===> 2            1 = BASIC, 2 = FAST
```

Improved Insert Performance in Data Sharing with RPN

- Issue
  - PBR table spaces with RPN may experience higher CPU for inserts vs a PBR with APN
    - Caused by a significant increase in false contentions for page P-locks
    - Primarily affects table spaces using row-level locking
      - Due to page P-locks for data pages
      - Db2 must generate more unique hash values for lock resources used in IRLM and XES for locking
- Db2 13 (M500)
  - Internal changes to resource hash values of page P-locks
  - New hash algorithm
    - Provides a more balanced distribution of resource hash values
    - Reduces false locking contentions and CPU processing impacts
  - Data page and index page header definitions have been modified
  - Large object (LOB) and XML pages are not affected by this change
  - PBR RPN table spaces created with a lower function level or in Db2 12
    - Must run REORG (entire tablespace) or LOAD REPLACE to use new lock hash values for PBR RPN



PBG to PBR Conversion

- Prior to 13
  - In order to convert a PBG to PBR several step must be taken along with an outage
- Db2 13 (M500)
  - Can convert a table's partitioning scheme from partition-by-growth (PBG) to partition-by-range (PBR) with minimal application impact
  - Use an ALTER TABLE statement with new ALTER PARTITIONING TO PARTITION BY clause
    - Conversion is immediate
    - If data sets are already defined, this becomes a pending change that is materialized by a REORG
  - Table space is converted to use relative page numbers (RPNs)
  - Conversion process handles any existing indexes on the table
    - Db2 does not change any aspects or attributes of those indexes
    - Consider creating partitioned indexes on the table after completing the conversion

```
ALTER TABLE TS1.TAB1
ALTER PARTITIONING TO PARTITION BY RANGE (COL1)
(PARTITION 1 ENDING AT (199),
PARTITION 2 ENDING AT (299),
PARTITION 3 ENDING AT (399),
PARTITION 4 ENDING AT (MAXVALUE));
```

Remove Stack Limits for Pending Alters

- Issue
  - When a PBG to PBR conversion is pending Db2 issues SQLCODE -20385 if some alterations are issued
    - Table space - BUFFERPOOL, DSSIZE, SEGSIZE, MEMBER CLUSTER
    - Table – ALTER COLUMN, DROP COLUMN
    - Index – BUFFERPOOL, COMPRESS
  - At least two executions of the REORG utility are required to complete changes needed for conversion
- Db2 13 (M500 PH51359 12/22)
  - Removed stacking limitations for PBG to PBR conversions
  - Supports *stacking* of certain pending DDL changes when converted from PBG to PBR
  - Many pending definition changes can now be issued together and materialized by one REORG
  - Useful if need to enlarge the partition data set sizes to accommodate the distribution of data into the partitions, alter columns to be used as partitioning keys, or alter other table space or index attributes

Object Level	**Supported stacked pending definition changes for PBG to PBR conversion**
Table space	* BUFFERPOOL * DSSIZE * SEGSIZE(not UTS conversion) * MEMBER CLUSTER
Table	* ALTER COLUMN * DROP COLUMN
Index	* BUFFERPOOL * COMPRESS

Max Partitions 254 Default Change

- Issue
  - In Db2 12(M504) when MAXPARTITIONS not specified, default is 256
    - Default DSSIZE is 4G regardless of page size
    - Avoided a risk of failure for existing statements
      - Where default data set size might be greater than 4G depending on page size
      - Statements might fail with -904 with reason code 00D70008
        - If data sets for table space are not in a SMS data class with EF/EA
- Db2 13 (M500)
  - CREATE TABLESPACE uses MAXPARTITIONS 254 by default
  - When MAXPARTITIONS 256 explicitly specified
    - Default DSSIZE varies from 4G to 32G depending on page size
  - With MAXPARTITIONS 254 as default
    - Consistent result regardless of whether MAXPARTITONS is explicitly specified
      - Default DSSIZE is always 4G

Page size	MAXPARTITIONS	DSSIZE
Any	1-254	4G
4K	255-4096	4G
8K	255-4096	8G
16K	255-4096	16G
32K	255-4096	32G

Increased Number of Open Datasets

- Issue
  - With large increases in data sets needed there is a growing need for a larger number of open datasets
    - Due to business consolidation, Db2 member consolidation, data growth, and conversion of segmented table spaces to PBGs
  - When DSMAX is reached (within 3%) datasets get closed
    - If reopened, can cause performance problems in applications
  - Db2 11(Db2 10 via APAR PM88166) increased the maximum number from 100,000 to 200,000
  - Limit of concurrent open data sets is due to
    - Required 31-bit DBM1 private memory per open data set
    - Amount of 31-bit private memory remaining in address spaces after subtracting required common memory

DSMAX

- Db2 13 and z/OS 2.5
  - z/OS 2.5 moves a portion of the control blocks for open data sets ATB
    - DSMAX increases from 200,000 to 400,000
    - Db2 can use the extra room to increase number of concurrent data sets or other Db2 activities (i.e. concurrent threads)
    - Can improve performance by decreasing the need to constantly open/close datasets

Improved DATA CAPTURE

- Prior to Db2 13
  - Db2 performs the following quiescing as part of the DATA CAPTURE alteration processing
    - Quiesces static packages that depend on the altered table
    - Quiesces cached dynamic statements that depend on the altered table
  - In HA environments, making time available to run the alteration can be challenging
  - Repeated executions of ALTER TABLE DATA CAPTURE might time out and fail
    - Due to continuous concurrent activities on the table
    - May result in not being able to enable or disable data replication in a timely manner
      - Or an application outage may be needed to complete alteration
- Db2 13 (M500)
  - DATA CAPTURE alteration no longer waits for dependent DML statements to commit
  - Operation can be run successfully even when static or dynamic DML is running concurrently against the table
  - ALTER TABLE DATA CAPTURE enhancement provides the following benefits
    - Static packages that depend on altered table are no longer quiesced
    - Cached dynamic statements that depend on altered table are no longer quiesced

Longer Column Names

TABLE_COL_NAME_EXPANSION

- Prior to Db2 13
  - Column names were limited to 30 bytes
- Db2 13 (M100)
  - Can define a column with a name longer than 30 bytes of EBCDIC, up to 128 bytes
  - TABLE_COL_NAME_EXPANSION – zparm controls whether column names can be longer than 30 bytes
    - OFF (default) - column names longer than 30 EBCDIC bytes must not be specified
    - ON - column names can be up to 128 EBCDIC bytes
  - When TABLE_COL_NAME_EXPANSION = ON
    - A column can be named, or reassigned a different name, longer than 30 EBCDIC bytes, and no warning is issued
      - Can occur in the following situations
        - Statements for tables, views, or indexes that define new columns or rename existing columns
        - INSERT, UPDATE, DELETE or MERGE statements define include columns
        - Common table expressions that explicitly specify column names
        - AS clause in SELECT or FROM clause of a query defines an alternate name for a column of a result table
  - Limitations when using SQLDA
    - Any interface using SQLDA to return column names will only return up to 30 bytes
      - SPUFI, QMF, DSNTEP1, DSNTIAUL, DCLGEN etc..

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 FL503 - APAR PI95480 (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 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

Non-Deterministic Expression for Column Auditing

PI95480, 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

Use of built-in Db2 support for audit columns 14% less CPU over a user defined trigger

Deleted row

ROW CHANGE TIMESTAMP – New Default

- Prior to Db2 13 - FL 503
  - Db2 derives default values for existing rows from page header from row
    - RBA for standalone Db2 subsystems
  - In data sharing, default based on an internal mapping table between LRSN and a timestamp
    - Inserts, deletes, or updates can change default ROW CHANGE TIMESTAMP value for unchanged rows
      - Can lead to unpredictable results
  - ROW CHANGE TIMESTAMP is often used in 'optimistic locking'
- Db2 13 - FL 503
  - New default values in existing rows for added ROW CHANGE TIMESTAMP columns
  - Db2 uses a constant default value when a new ROW CHANGE TIMESTAMP column is added
  - ALTER TABLEs with ADD COLUMN for ROW CHANGE TIMESTAMP columns
    - Sets value in DEFAULTVALUE column in SYSIBM.SYSCOLUMNS = timestamp of ALTER TABLE
  - CREATE TABLE processing to define a ROW CHANGE TIMESTAMP column does not set DEFAULTVALUE

ROW CHANGE TIMESTAMP

Movement from Segmented Multi-Table Tablespace to UTS

- Issue
  - Many segmented (non-UTS) table spaces exist with multiple tables
  - Multi-table segmented table spaces have now been deprecated
  - Need to move to UTS – which only supports one table per table space
- Db2 12 – FL 508
  - Support for moving tables from deprecated multi-table simple or segmented table spaces to PBG UTS
    - Without taking an outage
  - ALTER TABLESPACE with MOVE TABLE option
    - To move tables from a source multi-table table space to target partition-by-growth table spaces
  - Pending change is materialized by running the REORG utility on the source table space
  - Multi-step process (details in Db2 Administration Guide)
    - Identify tables to move and create target table space
    - Considerations for
      - Number of PBG parts (IBM recommends 1)
      - Size for size of initial partition (64 GB should be enough)
    - Identify packages that will become invalidation and plan rebinds accordingly

```
ALTER TABLESPACE ts1 MOVE TABLE tb1
TO TABLESPACE db1.ts2
```

Moving Multi-Table Tablespace to UTS - Steps

- Use ALTER TABLESPACE with MOVE TABLE option to move tables from a source multi-table table space to target PGB UTS
- If data sets of source table space are already created
  - MOVE TABLE becomes a pending definition change to be materialized by REORG on source
- Before moving any tables, complete the following tasks to prepare
  - Identify tables to move
    - Use this query to identify all multi-table table spaces and the number of tables in each table space:

```
SELECT CURRENT SERVER AS DB2LOC,
DBNAME, DBID, NAME AS TSNAME, NTABLES
FROM SYSIBM.SYSTABLESPACE
WHERE NTABLES > 1
ORDER BY DBNAME, TSNAME;
```

    - Use this query to identify all tables in a multi-table table space:

```
SELECT CURRENT SERVER AS DB2LOC,
TSP.DBNAME, TSP.DBID, TSP.NAME as TSNAME, TSP.NTABLES, TAB.CREATOR AS
TBCREATOR,TAB.NAME AS TBNAME, TAB.TYPE AS TBTYPE
FROM SYSIBM.SYSTABLESPACE AS TSP,
SYSIBM.SYSTABLES AS TAB
WHERE TSP.NTABLES > 1
AND TSP.DBNAME = TAB.DBNAME
AND TSP.NAME = TAB.TSNAME
ORDER BY TSP.DBNAME, TSP.NAME, TAB.CREATOR, TAB.NAME;
```

Moving Multi-Table Tablespace to UTS – Number of Tables

- Need to put some thought into how many tables to move at once
- Determine number of moved tables to materialize in a single REORG job
    - Can move only one table per ALTER TABLESPACE MOVE TABLE statement
        - Can materialize multiple pending MOVE TABLEs in a single REORG
    - Consider the following issues:
        - Processing time for ALTER TABLESPACE MOVE TABLE
            - $(n + 1) \times t$
        - Time to materialize pending definition changes via REORG
            - $2 \times (n + 1) \times t$
        - REORG and SWITCH phase for all shadow data sets

n - number of unmaterialized pending MOVE TABLEs
t - processing time for a single
ALTER TABLESPACE MOVE TABLE statement

Recommendation
Not to move more than a few hundred tables in single REORG!!!

Multi-Table Tablespace to UTS – Invalidations/REBINDs

- REORG materializing ALTER TABLESPACE MOVE TABLE invalidates packages dependent on tables
  - Before moving tables, identify packages that will be invalidated and prepare rebinds
  - To identify packages invalidated by materialization of a MOVE TABLE, run below query

```
SELECT DISTINCT DCOLLID, DNAME, DTYPE
FROM SYSIBM.SYSPACKDEP
WHERE BQUALIFIER = object_qualifier
AND BNAME = object_name
AND BTYPE = object_type
ORDER BY DCOLLID, DNAME;
```

object_qualifier - schema of table being moved
object_name - name of table being moved
object_type - **T -** Normal table; **M -** MQT

- Existing table-level statistics still persist after REORG
  - Minimal risk for access path regression from rebinds/autobinds of invalidated packages
- Take one of the following actions after REORG to further mitigate risk of access path regression:
  - Run a stand-alone RUNSTATS job
  - Rebind invalidated packages, with APREUSE(WARN) to minimize access path change
  - If enabled, an autobind also issues APREUSE(WARN)

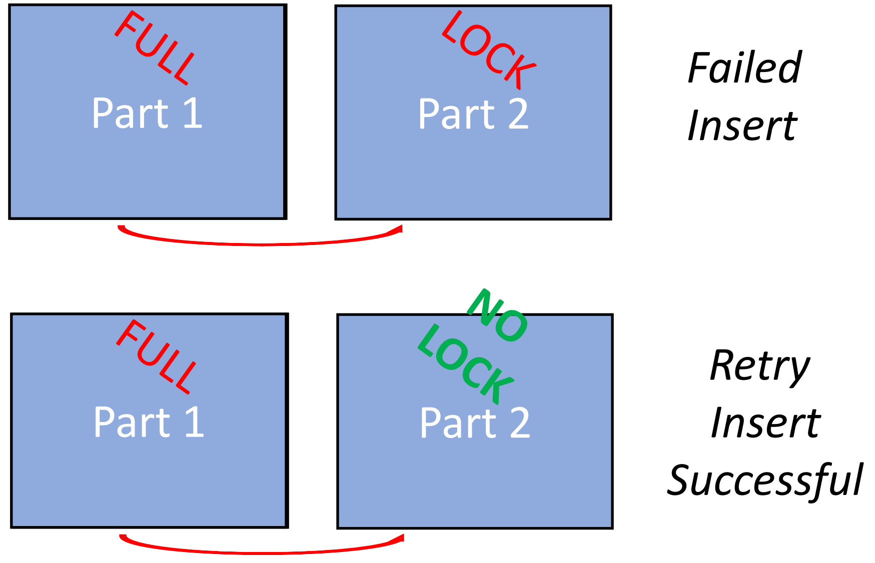
UTS PBR Tables Spaces and Hidden ROWID Support

- Issue:
  - No natural partitioning key
    - Leads to usage of PBGs for large table spaces with many partitions
    - Recommended to only use PBGs for older segmented table spaces, not large partitioned table spaces
  - May use ROWID (99.99% random generated value) as partitioning key
- Db2 12 – PI76972, PI77310, PI77302
  - Hidden ROWID support to partitioning key for PBR table spaces
    - Lessens need to use PBGs to support tables space w/o partitioning key
  - Benefits when no natural partitioning key exists
    - Allows table to be partitioned
    - Provide a better spread of data across partitions
  - Application transparency
  - Possible benefits ….
    - Improved insert throughput and less lock/latch contention on index and data
  - Possible problems…
    - Maintenance, clustering, sequential process, write I/Os, free space, inefficient index lookaside, etc..

```
CREATE TABLE TESTTAB
( COL1 CHAR(10) NOT NULL,

ROW_ID ROWID NOT NULL
IMPLICITLY HIDDEN GENERATED ALWAYS)

PARTITION BY (ROW_ID)
(PARTITION 1 ENDING AT (X'0FFF'),
….
PARTITION 4 ENDING AT (X’3FFF'),
PARTITION 5 ENDING AT (MAXVALUE))
```

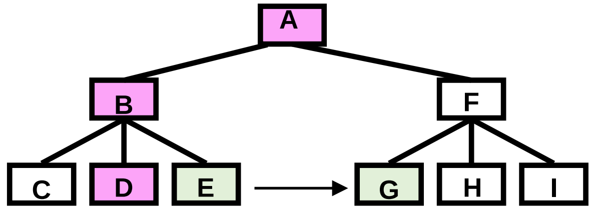
Faster PBG Insert

- Db2 12
  - During an INSERT process, Db2 only performs a single attempt to obtain a lock on target partition
    - If attempt failed, target partition was skipped, and next partition was evaluated
    - This process would continue until INSERT either successfully obtained a partition lock
      - Or it finished searching all existing partitions without obtaining a partition lock
  - In most cases, duration of partition lock contention is short
    - However, INSERT was terminated
      - INSERT did not make another attempt to obtain a lock on a partition after first failed attempt
- Db2 13 (M100)
  - Improves cross-partition search algorithm for INSERT transactions
  - Enhancements to partition retry logic after a lock contention and to bidirectional partition search logic
  - Introduces retry logic for INSERT operations
  - An extra attempt is made to obtain a partition lock on a PBG table space after a failed first attempt
    - Increasing the success rate of INSERT operations
  - Improves performance without changing how Db2 is configured, monitored, and used



Improved Index-Lookaside

- Prior to Db2 13
  - Index accesses for INSERT and DELETE operations can use index look-aside for
    - Clustering indexes
    - Non-clustering indexes with high cluster ratios based on catalog statistics
  - Index look-aside is not supported for UPDATEs



- Db2 13(M100)
  - Improvement to index look-aside by storing last accessed index non-leaf and leaf page information for INSERT, DELETE, and UPDATE
    - When thread does more than three INSERT, DELETE, and UPDATE operations in same commit scope
    - Regardless of cluster ratio or inaccurate catalog statistics
  - Benefits FTB because it reduces index GETPAGE cost for INSERT, DELETE, and UPDATE
  - Dynamically adjusts to avoid any impact when INSERT, DELETE, and UPDATE pattern is random and index look-aside does not provide a benefit
  - Can expect more benefits when fast index traversal is not used for these indexes
  - Greater reduction in GETPAGEs for non-clustering indexes, especially when they have a high cluster ratio
  - Helps reduce cost for maintaining indexes during INSERT, UPDATE and DELETEs due to reduced GETPAGES on index
    - Overhead usually runs about 25% for each index

FTB Support of Non-Unique Indexes

- Issue
  - FTBs can significantly reduce get pages and CPU during index-tree traversals
  - Many restrictions on which indexes qualified for their use
    - Be UNIQUE (INCLUDE supported)
    - Have a key size 64 bytes or less
    - Not have multiple versions
- Db2 12 FL 508 PH30978
  - Fast index traversal (FTB) support for non-unique indexes
  - Supports fast index traversal (FTB) for non-unique indexes
  - To enable use of FTBs for non-unique indexes
    - Set value of FTB_NON_UNIQUE_INDEX subsystem parameter to YES
  - Only key size of ordering columns had to be 64 bytes or less
    - Columns in the INCLUDE list no longer counted toward size limit for index key
    - However, fast index traversal was not used for columns in INCLUDE list
  - Setting value FTB_NON_UNIQUE_INDEX = YES, all non-unique indexes became eligible for fast traversal
    - But, like for unique indexes, key size for columns of non-unique indexes had to be 56 bytes or less

FTB_NON_UNIQUE_INDEX

Fast Traverse Block - Eligibility Improvements

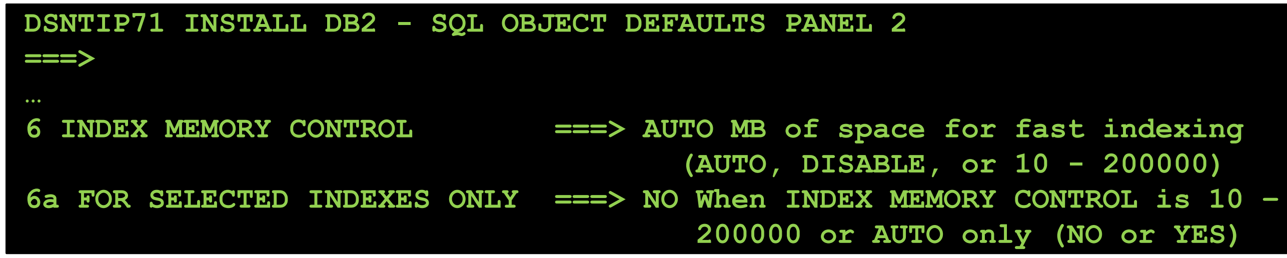
- Prior to Db2 13
  - Only key size of ordering columns had to be 64 bytes or less
    - Columns in the INCLUDE list no longer counted toward size limit for index key
    - However, fast index traversal was not used for columns in INCLUDE list
  - APAR PH30978 strengthened FTB support for non-unique indexes
  - Setting FTB_NON_UNIQUE_INDEX = YES, all non-unique indexes are eligible for fast traversal processing
    - However key size for columns of non-unique indexes had to be 56 bytes or less
- Db2 13 (M500)
  - Continues to enhance FTB and extends the maximum key length for FTB eligible indexes as follows:
    - For unique indexes, the key length is limited to 128 bytes
    - For unique indexes with INCLUDE columns, the unique part of the index must not exceed 128 bytes
    - For non-unique indexes, the key length is limited to 120 bytes
  - FTB_NON_UNIQUE_INDEX zparm (YES default)
  - Non-unique indexes are automatically eligible for FTB processing
    - If they meet other eligibility criteria, such as the maximum key lengths

FTB_NON_UNIQUE_INDEX

FTB Usage Recommendation – Selective Index Usage

- Db2 12
  - FTBs use controlled by Db2 through automation
  - Enabled at subsystem level
- Db2 12 - PH23238
  - Enables of use fast index traversal (FTB) for selected indexes
  - Enables focused testing in preparation to exploit FTBs
  - FOR SELECTED INDEXES ONLY = YES - DSNTIP71
  - Enhancement to INDEX_MEMORY_CONTROL zparm
    - INDEX_MEMORY_CONTROL=(SELECTED, AUTO)
      - SELECTED ‘n’ - storage (10 - 200000 MB)
  - Controls specific indexes to be used
    - ACTION='A' in SYSIBM.SYSINDEXCONTROL

INDEX_MEMORY_CONTROL



```
DSNTIP71 INSTALL DB2 - SQL OBJECT DEFAULTS PANEL 2
===>
…
6 INDEX MEMORY CONTROL        ===> AUTO MB of space for fast indexing
                                     (AUTO, DISABLE, or 10 - 200000)
6a FOR SELECTED INDEXES ONLY  ===> NO When INDEX MEMORY CONTROL is 10 –
                                      200000 or AUTO only (NO or YES)
```

SSID	PARTITION	IXNAME	IXCREATOR	TYPE	ACTION	MONTH_WEEK	MONTH	DAY	FROM_TIME	TO_TIME
DB2P	5	IX1P	DBA1	F	F	W		5		
DB2T	4	IX1T	DBA1	F	F	W				
DB2Q	1	IX1Q	DBA1	F	D	M		1		0100

In Memory Indexes – Usage Recommendation (cont..)

- Be current on maintenance
  - PH16429, PH19484, PH21916, PH24667, PH25240, PH25801, PH26109
- Only turn it on specifically for index(es) of interest PH23238/UI69968
  - Not globally by default
- SYSINDEXCONTROL - Use of memory for FTB allocated for index
  - IXNAME – Index to use FTB
  - TYPE - Purpose for which memory is used
    - F - Fast Index Traversal(FTB)
  - ACTION – Action being performed
    - F - Force FTB creation
    - D - Disable FTB creation
    - A - Automatic FTB creation

IBM NOTE (12/9/2020):
1.Use the index-level granularity provided by APAR PH23238 / UI69968 to activate FTBs in selected cases
2.Then as results of testing and usage warrant, consider whether environment is best served by maintaining index-level control or could benefit from reactivation of the INDEX_MEMORY_CONTROL subsystem parameter

SYSINDEXCONTROL

SSID	PARTITION	IXNAME	IXCREATOR	TYPE	ACTION	MONTH_WEEK	MONTH	DAY	FROM_TIME	TO_TIME
DB2P	5	IX1P	DBA1	F	F	W		5		
DB2T	4	IX1T	DBA1	F	F	W				
DB2Q	1	IX1Q	DBA1	F	D	M				

DISPLAY STATS - INDEXTRAVERSECOUNT

- Db2 12 PH34859 (5/21)
  - Display information about use of fast index traversal (FTB) with DISPLAY STATS
  - Can issue a DISPLAY STATS command and specify new INDEXTRAVERSECOUNT option
    - Displays information about the use of fast index traversal (FTBs) for a specific index, or a list of the first *n* indexes with the most uses of FTBs
  - Db2 returns the results in message DSNT830I

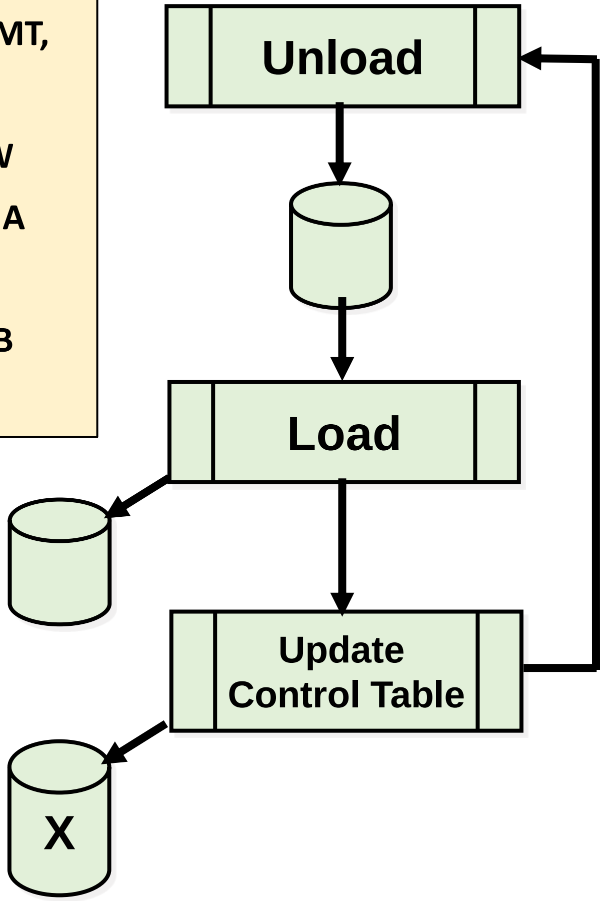
```
-DISPLAY STATS(INDEXTRAVERSECOUNT) DBNAME(DB1) SPACENAM(IX1) PART(1)
```

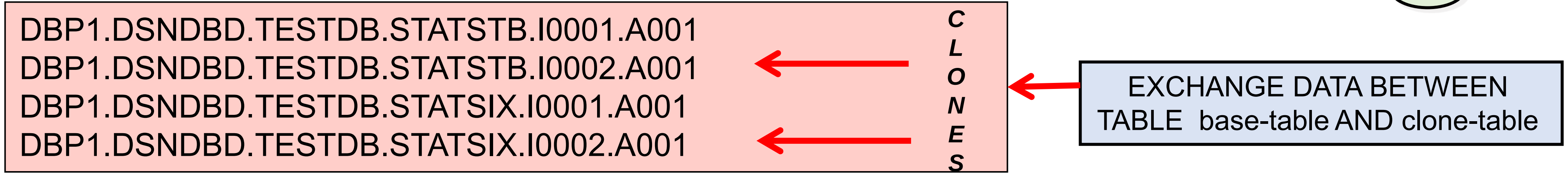
```
DBID   PSID    DBNAME     IX-SPACE    LVL  PART   TRAV. COUNT
------ ------  ---------- ----------  ---  ----   ----------------
 0017  0005     DB1        IX1        003  0001   00000000999

******* DISPLAY OF STATS ENDED ******************
```

Mirror and Clone Tables – Past Solutions

- Multiple summary user-table solution
  - Two identical summary tables
  - Control (switch) table facilitates switching
  - User defined 'automated' process does the refresh
    - UNLOAD detail data with summary query
    - LOAD summary data to unused summary table
    - Use the control table to switch tables
- Clone tables
  - Db2 maintained clone tables
  - Can update primary then switch to clone
  - EXCHANGE can be an outage depending on usage

```
SELECT  COALESCE(A.TOT_AMT,
B.TOT_AMT)
FROM SWITCH_TABLE AS SW
LEFT JOIN  SUMMARY_A AS A
ON SW.ACTIVE_TABLE = 'A'
LEFT JOIN SUMMARY_B AS B
ON SW.ACTIVE_TABLE = 'B';
```





```
DBP1.DSNDBD.TESTDB.STATSTB.I0001.A001
DBP1.DSNDBD.TESTDB.STATSTB.I0002.A001
DBP1.DSNDBD.TESTDB.STATSIX.I0001.A001
DBP1.DSNDBD.TESTDB.STATSIX.I0002.A001
```

EXCHANGE DATA BETWEEN TABLE base-table AND clone-table

Online Load Replace

LOAD REPLACE SHRLEVEL REFERENCE

- Issue
  - When loading a table, it is unavailable to applications
  - Past designs could use a ‘mirror table/switching’ technique or clone table/exchange functionality
    - Either method had its own challenges
      - Mirror table method required a control table and additional SQL
      - Clone tables had many restrictions and the exchange could be long depending on usage

- Db2 12 (PI69095, Db2 11 PI67793)
  - New function of LOAD REPLACE SHRLEVEL REFERENCE for LOAD utility
    - Provides concurrent read application access to the target table
      - While new data is being loaded into a set of shadow table data sets
    - A new SWITCH phase is introduced to LOAD utility
      - To switch access between original and shadow table data sets

System Temporal and Archive Enabled Tables – Trigger Issue

- Cannot not create, alter, or rebind a trigger package
  - If system time temporal or archive enabled tables were referenced in WHEN clause
  - Or if trigger definition or bind options were specified YES for SYSTEM_TIME SENSITIVE or ARCHIVE SENSITIVE
- Time travel (or row versioning) and archive transparency features were restricted in trigger WHEN clauses

SYSTEM TEMPORAL

```
CREATE TABLE CUSTOMER
(CUST_ID INT NOT NULL,
 START_TSP TIMESTAMP NOT NULL GENERATED ALWAYS AS ROW BEGIN,
 END_TSP TIMESTAMP NOT NULL GENERATED ALWAYS AS ROW END,
 DATA_COL CHAR(10),
 PERIOD SYSTEM_TIME(START_TSP,END_TSP));

CREATE TABLE CUSTOMER_HIST_TABLE
(CUST_ID INT NOT NULL,
 START_TSP TIMESTAMP NOT NULL ,
 END_TSP TIMESTAMP NOT NULL ,
 DATA_COL CHAR(10));

ALTER TABLE CUST_TABLE ADD VERSIONING
USE HISTORY TABLE CUST_HIST_TABLE;
```

ARCHIVE ENABLED

```
CREATE TABLE CUSTOMER
(CUST_ID CHAR(10) NOT NULL,
AMOUNT INT NOT NULL);

CREATE TABLE CUSTOMER_ARCHIVE
(CUST_ID CHAR(10) NOT NULL,
AMOUNT INT NOT NULL);

ALTER TABLE CUSTOMER ENABLE ARCHIVE
USE CUSTOMER_ARCHIVE;
```

```
CREATE TRIGGER CUSTACCU1
AFTER UPDATE ON ACCOUNT
REFERENCING OLD AS
OLDPERSON
FOR EACH ROW MODE DB2SQL
WHEN (EXISTS
(SELECT 1
   FROM CUSTOMER
  WHERE  (CUST_ID,
STRT_DT…
```

Trigger NOT allowed due to WHEN reference system temporal or Archive enabled table

Triggers, Temporal and Archive Transparency

- Issue
  - Cannot create a trigger with WHEN clause containing a query referencing a system-period temporal or archive-enabled table
- Db2 12 (FL 505)
  - Can reference system-period temporal and archive-enabled tables in WHEN clauses of basic and advanced triggers
    - Regardless of trigger definition for SYSTEM_TIME SENSITIVE or ARCHIVE SENSITIVE
    - Or settings of the SYSTIMESENSITIVE or ARCHIVESENSITIVE bind options
  - Time machine (row versioning) and transparent archive data retrieval are fully supported in WHEN clause
- When creating or altering an advanced trigger
  - Can specify SYSTEM_TIME SENSITIVE YES(default) for system-time temporal tables
  - Or ARCHIVE SENSITIVE YES (default) for archive-enabled tables
  - Can be ALTERed for advanced triggers

SYSTEM_TIME SENSITIVE

ARCHIVE_SENSITIVE

Real-Time Statistics Scalability Improvements

- Prior to 13
  - As data volumes become larger, the widths of some columns in the real-time statistics tables are not large enough to accommodate larger values
  - During high volume processing, lock escalation could occur on the real-time statistics history table spaces
    - Negatively affect concurrency and performance

- Db2 13 (M501)
  - Column data types are changed BIGINT instead of INTEGER, or INTEGER instead of SMALLINT for several key columns
    - Values such as REORGINSERTS
  - Lock escalation is disabled on the following table spaces:
    - DSNDB06.SYSTSTSS and DSNDB06.SYSTSISS for the RTS tables
    - DSNDB06.SYSTSTSH and DSNDB06.SYSTSISH for the RTS history tables

LOB Compression Improvements

- Db2 zOS LOB Compression
  - LOB compression in Db2 is not dictionary-based like table space compression
  - Primary goal of compression is to save disk space
  - There is overhead for compressing LOB data during INSERTs and overhead for decompressing during SELECTs
    - Good candidates are LOB columns that store data in formats such as .docx, .txt, XML, and HTML
    - No benefit for binary files having a low compression ratio, therefore Db2 does not compress the data
  - DSN1COMP estimates compression ratio and number of data pages saved before compression of LOB is implemented
- Prior to Db2 13
  - On IBM zEC12, z13 and z14, IBM zEnterprise® Data Compression(zEDC) express card is required for LOB compression
- Db2 13 and IBM z15 - Integrated Accelerator for zEDC
  - IBM z15 and later use the Integrated Accelerator for zEDC to perform compression
    - Integrated Accelerator for zEDC replaces the existing zEDC Express adapter card in the I/O drawer
  - IBM z15 uses one accelerator chip per processor, and it is shared by all processor cores
    - Offers low latency and high bandwidth for compression
  - Improves compression ratio across well-known file types such as txt, pdf, jpg, png, ppt, tif.
    - Can achieve up to 70% faster compression/decompression operations in some cases
    - Reduces CPU cost and can handle much more compression work than PCIE card on IBM z14 hardware

Transparent Data Encryption

- Db2 12 (M502)
  - Provides additional function
  - Encryption exploits a hardware instruction executed in an attached Crypto processor, minimizing cost of instruction
  - Implemented in media manager, a low-level access mechanism (part of DFSMS) used by BSAM, QSAM, VSAM, and Db2
    - Does not support tape
  - z/OS allows three methods of enabling encryption(in priority order)
    - In a RACF dataset profile
    - Attribute of dataset (in JCL with DSKEYBL or KEYLABEL in IDCAMS)
    - Attribute in a SMS DATACLAS
    - To be encrypted, a dataset must have Extended attribute
    - SMS encryption available in z/OS 2.2 w/PTF for APAR OA50569 or z/OS 2.3
  - Methods of enabling encryption for Db2 datasets supported in Db2 11/12
    - Will need attached Crypto processor
  - Can be used for image copies and utility work files as early as Db2 11
  - Available through DDL for Db2 data in Db2 12 (M502)
    - Allow "KEY LABEL" as part of CREATE/ALTER TABLE and STOGROUP
    - Archive and history tables must be encrypted separately
    - After encryption has been enabled for a tablespace/index, next Reorg of object will encrypt data

```
CREATE TABLE EMP_TAB
(DEPTNO CHAR(3) NOT NULL,
DEPTNAME VARCHAR(36) NOT NULL,
MGRNO CHAR(6) ,
ADMRDEPT CHAR(3) NOT NULL,
LOCATION CHAR(16) ,
PRIMARY KEY(DEPTNO) )
IN DSN8D12A.DSN8S12D
KEY LABEL EMPKEYLABEL
```

Functions - ENCRYPT_DATAKEY – DECRYPT_DATAKEY

- ENCRYPT_DATAKEY (FL505)
  - Returns a value that is result of encrypting first argument using specified key label and algorithm

Ex: Encrypt values in character column SSN and insert resulting values into PROTECTED_SSN column in CUSTOMER table: PROTECTED_SSN column is defined as VARBINARY. Use 'MYKEYLABEL' as key label, and AES256D encryption algorithm.

```
UPDATE CUSTOMER SET PROTECTED_SSN = ENCRYPT_DATAKEY(SSN,'MYKEYLABEL', AES256D)
```

- DECRYPT_DATAKEY (FL505)
  - Returns a value that is result of decrypting first argument (previously encrypted using ENCRYPT_DATAKEY function) using algorithm that was specified when the data was encrypted

Ex: Decrypt value in PROTECTED_SSN VARBINARY column. Encrypted data was originally a character string.

```
SELECT DECRYPT_DATAKEY_VARCHAR(PROTECTED_SSN) FROM CUSTOMER WHERE CID = ?
```

IBM z15 Huffman Compression

TS_COMPRESSION_TYPE

- Db2 12 - FL509
  - Introduces support for compression of Db2 data with IBM Z hardware-based entropy encoding (Huffman) compression with the IBM z14® Compression Coprocessor (CMPSC)
  - APAR PH04424 delivered the functional code that supports IBM z14 Huffman compression
  - TS_COMPRESSION_TYPE subsystem parameter
    - Controls compression method for Db2 subsystems on IBM z14 hardware with Huffman compression enabled
    - Huffman compression is only available for data in UTSs
      - Regardless of the setting of the TS_COMPRESSION_TYPE
  - Turned on at object level
    - COMPRESS YES HUFFMAN | FIXEDLENGTH in DDL
  - Status can be seen in catalog
  - DSN1COMP can give stats for fixed length to Huffman
  - Does not support partial decompression
    - Can have effect on query performance

SYSTABLEPART
COMPRESS_USED - F, H, ‘ ’ or NULL
COMPRESS - Y, F, H, ‘ ’

SYSTABLESPACE
COMPRESS - Y, F, H, ‘ ’ or NULL

CREATE/ALTER TABLESPACE….COMPRESS YES HUFFMAN

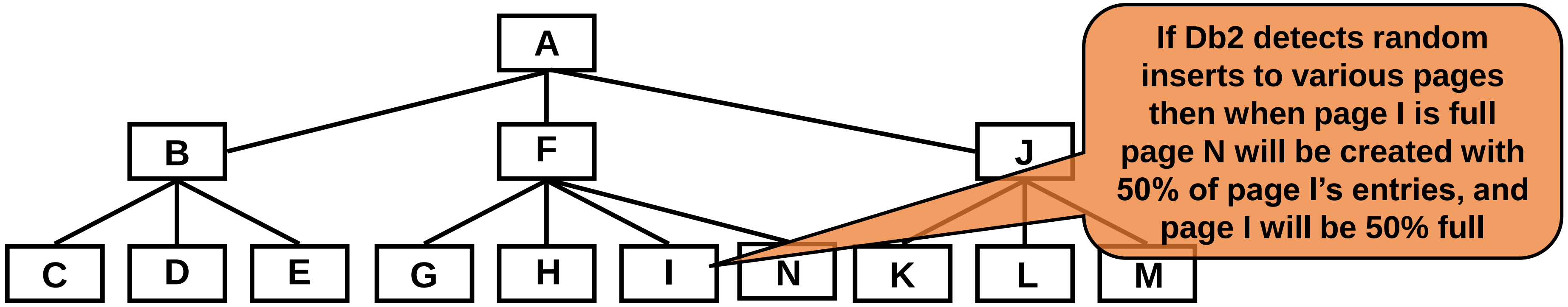
DSN1COMP Usage for FIXED vs HUFFMAN

	UNCOMPRESSED	Estimated state Compressed FIXED	Estimated state Compressed HUFFMAN	Calculated Compressed from INPUT DICTIONARY
DATA (IN KB)	329,777	136,431	119,264	143,008
PERCENT SAVINGS		58%	63%	56%
AVERAGE BYTES PER ROW	163	69	61	72
PERCENT SAVINGS		57%	62%	56%
DATA PAGES NEEDED	91,305	38,182	33,334	39,623
PERCENT DATA PAGES SAVED		58%	63%	56%
DICTIONARY PAGES REQUIRED	0	32	32	32
ROWS SCANNED TO BUILD DICTIONARY		921	921	N/A
ROWS SCANNED TO PROVIDE ESTIMATE		2,100,000	2,100,000	N/A
DICTIONARY ENTRIES		4,096	4,080	4,096
TOTAL PAGES (DICTIONARY + DATA)	91,305	38,214	33,366	39,655
PERCENT SAVINGS		58%	63%	56%

COMPTYPE (ALL) - specifies that DSN1COMP is to report the estimated space savings that are to be achieved by both Huffman compression and fixed-length compression

Improved Index Page Split Info

- Issue:
  - When data is inserted into a base table, corresponding indexes are modified accordingly
  - If there is not enough space on the page to accommodate the change the index page may split
    - Causes performance issue for inserts and synchronous I/Os for GBP dependent index page sets
- Prior to Db2 13
  - IFCID 0359 records contain information index split events
    - However, the information that is recorded is not detailed enough to identify the cause of performance issues
    - Is also disabled by default, and can miss capturing some abnormal index split situations
      - Enabled under statistics trace class 3 and performance trace class 6
      - Record is generated when an index split is considered an abnormal split process
        - i.e. total elapsed time is greater than 1 second
      - Contains the DBID, PSID, member ID, URID, page number, start/end timestamp of split


A
B
F
J
C
D
E
G
H
I
N
K
L
M
If Db2 detects random inserts to various pages then when page I is full page N will be created with 50% of page I's entries, and page I will be 50% full

Improved Index Page Split Info (cont..)

- Db2 13 (501)
  - New RTS (Real Time Statistics) columns in SYSIBM.SYSINDEXSPACESTATS catalog table are populated
  - Records and aggregate general index split information since last table reorganization, index rebuild, or load replace
  - Db2 starts populating these RTS columns as soon as the catalog level V13R1M501 update completes

Column name	Data type	Description
REORGTOTALSPLITS	INTEGER	The number of index splits since last reorganization or rebuild.
REORGSPLITTIME	BIGINT	Aggregated elapsed time for all index splits since last reorganization or rebuild.
REORGEXCSPLITS	INTEGER	The number of abnormal index splits (such as elapsed times greater than 1 second) since last reorganization or rebuild.

Deletion of Old Statistics When Using Profiles

- Issue
  - Statistics can become stale and inconsistent
  - Occurs overtime with RUNSTATS executions with various options or manual catalog updates
- Db2 12 (FL 507 – PH16345)
  - Deletion of old statistics when using profiles
  - When executing RUNSTATS with USE PROFILE option
    - Db2 collects only those statistics that are included in specified profile
    - Deletes existing statistics that are not part of profile for target object
      - Frequency
      - Key cardinality
      - Histogram statistics
  - Removes stale distribution statistics without impacting concurrently running dynamic SQL
  - Applies when profiles are used to gather inline statistics by REORG TABLESPACE and LOAD

SQL and Applications

Rebind Phase-In

- Issue
  - If a package is “in-use” it cannot rebind
    - A package using RELEASE(DEALLOCATE) allocated with a persistent thread is considered “in-use” until it is terminated
    - Packages executed frequently may also never reach use count of zero (regardless of RELEASE option)
    - Attempt to break-in may time out or could be disruptive
- Db2 12 - FL505 (PH12186)
  - Rebind phase-in support for packages that are being used for execution
  - Allows Db2 to rebind a package concurrently with execution of the package
  - Rebind operation creates a new copy of package
    - When rebind operation finishes
      - New threads can use new package copy immediately
        - Existing threads continue to use copy that was in use prior to rebind (phased-out copy) without disruption
  - Db2 waits for duration set by IRLMRWT before creating package copies for rebind phase-in operation
  - Enables switching to previous access paths and runtime structures gradually
    - Switch to previous access path is phased-in
    - Allows regression recovery without incurring an application outage

PLANMGMT(EXTENDED)
APREUSESOURCE(CURRENT

REBIND Phase-in - Concurrency Issue

- Issue:
  - Cannot issue REBIND of an application package because that could cause later transaction waiting and timeout on package lock
  - Transactions starting slightly after REBIND of package could experience performance slow down and possibly timeout on package lock
  - Due to the fact that REBIND requests an SIX lock on the package
    - Which is incompatible with transactions requesting an S lock on package
  - Can result in -913|-911

- Db2 12 (FL505) PH28693
  - Introduces a concurrency improvement for REBIND phase-in
  - Removes conditional SIX lock
    - Instead it will only obtain a U lock on the package
      - Allows subsequent transactions executing that package to run in parallel
        - Because U lock is compatible with a transaction's S lock

REBIND Phase-In – Storage Issue

- Issue
  - Every REBIND will create and persist a “phased-out” copy
    - Regardless of whether package is active or not
  - Space requirement for extra package copy in SYSPACKCOPY
  - Process required to FREE “phased-out“ copies and free up the respective copy ID for reuse
  - In data sharing REBIND PHASE-IN function gets storage for locking control blocks in ECSA but does not free it
  - Fix was to use PLANMGMT BASIC or OFF rather than EXTENDED
  - Recycling Db2 for z/OS would also free the storage (not really a good option)
- Db2 12 (FL505) PH35513
  - Db2 is modified to use existing storage
- Db2 12 APAR PH33295
  - New options to free unused phased-out copies
  - Reduces space in Db2 directory and catalog
  - PHASEOUT option on FREE Package
    - Frees eligible phased-out copies from directory, catalog, and access path repository
    - Phased-out package copies can be freed while applications that using package are running
    - Can also be freed regardless of the INVALIDONLY option
  - FREE PACKAGE PLANMGEMTS(INACTIVE)
    - Can be used to free eligible phased-out copies

```
FREE PACKAGE PLANMGMTSCOPE(PHASEOUT)
```

Profile Table Enhancement

- Prior to Db2 13
  - System profiles tables are used to monitor and control various aspects of a Db2 subsystem in specific application contexts
  - Set of criteria that identifies a specific context on a Db2 subsystem
    - Threads, connections, or application plans and packages that contain SQL statements with certain attributes
  - Only available for remote applications

- Db2 13 (M500)
  - Available for local applications

Controlling RELEASE of Packages via Profile Tables

- Issue:
  - Making database schema changes is challenging when online transactions are continuously running
  - DDL statements such as ALTER or DROP can impact applications that reference same
  - Applications performing SELECT, INSERT, UPDATE, and DELETE, depend on referenced database objects
  - To protect data integrity, applications must be serialized against the DDL process through a package lock
  - Amount of time that the application threads holds a package lock is controlled by RELEASE bind option
    - RELEASE(COMMIT) - package to be released at end of transaction
    - RELEASE(DEALLOCATE) - remain with running thread until it is deallocated
  - When DDL must be executed, high-performance DBATs must be stopped or deallocated
- Db2 13 (M500)
  - Increases likelihood of DDL success by providing new RELEASE_PACKAGE keyword in profile tables
  - Provides more control over managing packages bound RELEASE(DEALLOCATE) before attempting DDL changes
  - DSN_PROFILE_TABLE
    - Used to define profiles and filtering criteria to identify threads, connections, or packages
  - DSN_PROFILE_ATTRIBUTE_TABLE
    - Used to force Db2 to switch from using RELEASE(DEALLOCATE) to RELEASE(COMMIT) when profile starts

RELEASE_PACKAGE

Application Granularity for Locking Limits

- Issue
  - Lock limits are controlled by ZPARMs that are enforced the same for the entire subsystem
    - NUMLKUS – Locks per user
    - NUMLKTS – Locks per table space

MAX_LOCKS_PER_TABLESPACE
MAX_LOCKS_PER_USER

- Db2 12 (FL507)
  - Two new built-in global variables introduced to support application granularity for locking limits
  - MAX_LOCKS_PER_TABLESPACE
    - Maximum page, row, or LOB locks an application can hold simultaneously in a table space
      - 0- 104857600, NULL, DEFAULT
    - If application exceeds maximum number of locks in a single table space, lock escalation occurs
    - Corresponds to NUMLKTS
  - MAX_LOCKS_PER_USER
    - Max number of page, row, or LOB locks a single application can concurrently hold for all table spaces
      - 0- 104857600, NULL, DEFAULT
    - For table spaces defined LOCKSIZE PAGE, LOCKSIZE ROW, or LOCKSIZE ANY
    - Corresponds to NUMLKUS

```
SET SYSIBMADM.MAX_LOCKS_PER_TABLESPACE = 100
```

```
SET SYSIBMADM.MAX_LOCKS_PER_USER = DEFAULT
```

Application Level Timeout Control

- Prior to Db2 13
  - Timeout value for locks is a system-wide setting
  - All application locks are subject to this value (default 60 seconds)
  - Was not online changeable (must bounce Db2 thus outage)

IRLMRWT

- Db2 13 (M500)
  - Provides application controlled timeout setting
    - CURRENT LOCK TIMEOUT
  - New SPREG_LOCK_TIMEOUT subsystem parameter
    - Max value specified for SET CURRENT LOCK TIMEOUT statement
    - Default value for SPREG_LOCK_TIMEOUT_MAX is -1
      - Allows all valid values to be set for CURRENT LOCK TIMEOUT

SPREG_LOCK_TIMEOUT

SET CURRENT LOCK TIMEOUT = 30

Deadlock Priority Control

- Prior to Db2 13
  - When different lock requests from multiple threads a deadlock occurs
    - IRLM resolves deadlock by choosing a *victim* whose request is then denied
  - Application cannot influence which thread is chosen as deadlock victim
  - Batch data definition (DDL) jobs frequently fail due to deadlocks
    - Failing the DDL statements instead of other SQL is not optimal
  - Restart or retry logic and careful scheduling can help
    - Restart/retry logic requires code change and additional processing and is often not done
    - Scheduling is Increasingly difficult to optimize in continuous availability environments
- Db2 13 (M501)
  - New DEADLOCK_RESOLUTION_PRIORITY built-in global variable
  - Allows for specification of a deadlock resolution priority value to use in resolving deadlocks with other threads
    - 0 – 255, NULL, DEFAULT
      - Higher value = less likely lock requests by application will be the victim in a deadlock situation

DEADLOCK_RESOLUTION_PRIORITY

```
SET SYSIBMADM.DEADLOCK_RESOLUTION_PRIORITY = 100
```

Lock Avoidance for Singleton SELECT

- Issue
  - A *singleton SELECT* is a cursor SELECT that returns at most one row
- Db2 12(100) – PH49335 (Dec 2022)
  - Capability to use lock avoidance for singleton SELECT..ISOLATION(CS)..CURRENTDATA(YES)
  - New ZPARM - LA_SINGLESEL_ISOCS_CDY
    - YES – allow lock avoidance
  - Can significantly lessen locking activity for those statements
    - However, occasional false warning or error conditions are also possible during certain small timing windows
      - Such as SQLCODE -811 (more than one row returned) or SQLCODE +100 (no row found)
    - LA_SINGLESEL_ISOCS_CDY = YES
      - Should only be used if environment can tolerate these situations

LA_SINGLESEL_ISOCS_CDY

Greater Number of Concurrent Threads – Utilizing ATB Storage

- Before Db2 13
  - Large amount of agent local BTB storage that is consumed per thread
  - System can run low on BTB storage with large number of concurrent users
  - Largest user of local BTB storage is SQL text used for PREPARE and EXECUTE IMMEDIATE
    - SQL text can be as large as 2 MB
  - Dynamic SQL statements can also be nested due to triggers, UDFs, and stored procedures
  - Db2 keeps a copy of dynamic SQL text and attribute string in agent local BTB storage
    - During PREPARE and EXECUTE IMMEDIATE executions
- Db2 13 (M100)
  - Moving local BTB storage to ATB lifts local agent storage constraints
    - Allowing larger number of concurrent users
  - Statement text and attribute string for PREPARE and EXECUTE IMMEDIATE stored ATB
  - Db2 also optimizes interface between DBM1 and DIST address spaces
    - By allocating storage in shared ATB storage, thus avoiding cross-memory operations

BTB Storage Reduction – Application Impact - Example

Table (T1) with After-Insert Trigger - calls native SQL Stored Procedure → INSERT fires trigger which calls SP → SP issues dynamic SQL UPDATE

Assuming variables that hold SQL text are 2MB

Before 13

Prepare of INSERT acquires BTB storage to fit SQL and control Structures

Stored procedure prepares UPDATE and allocated BTB storage for 2MB variable

In Db2 13

Prepare of INSERT acquires ATB storage to fit SQL and control Structures

Stored procedure prepares UPDATE and allocated ATB storage for actual length of text

Sort Performance Improvement with Long Varchars

- Issue
  - If last column in a sort key is a long VARCHAR column
    - Sort allocates storage for the maximum column length that is defined
- Db2 13
  - If a query requires a sort for an ORDER BY and they sort key column is greater than 100 bytes
    - During the first execution
      - Sort records the maximum length for the long VARCHAR column
    - During subsequent executions
      - Sort allocates an amount of storage based on maximum actual data length determined in first execution
  - This reduces sort storage and work file usage in subsequent executions of the same query
  - Results in more CPU time and elapsed time savings
- The larger difference between actual data length and defined length
  - More memory storage and work file usage reduction for qualified queries can be realized

```
SELECT COL1, COL2, VARCOL1
FROM TAB1
ORDER BY COL1, VARCOL1;
```

IBM Db2 13 Performance Redbook SG248536

IBM results:
Class 2 CPU time - 10%
Sort buffer pool getpage requests - 24%

List Prefetch for MERGE

- Issue
  - The MERGE statement could not use an index which has any index key columns that are updated by the MERGE statement
  - MERGE statements would likely resort to table space scans
- Db2 13 (M500) PH47581 (9/22)
  - Introduces list prefetch as a possible access path for the MERGE statement
  - Index access with list prefetch can sometimes be used when an index column is being updated by MERGE statement
  - MERGE statements may choose index access with list prefetch as a potential access path
    - PREFETCH='L' in PLAN_TABLE

```
EXEC SQL
MERGE INTO CUSTTARG AS T1
USING CUSTSOUR AS T2
ON T1.C_CUSTKEY = T2.C_CUSTKEY
WHEN MATCHED
THEN
UPDATE SET T1.C_NAME = T2.C_NAME
WHEN NOT MATCHED
THEN
INSERT(C_CUSTKEY,C_NAME,C_ADDRESS,C_NATIONKEY,C_PHONE)
VALUES(C_CUSTKEY,C_NAME,C_ADDRESS,C_NATIONKEY,C_PHONE);
EXEC SQL COMMIT;
```

SELECT INTO Support for OPTIMIZE FOR n ROWS

- Prior to Db2 13 FL 503
  - FETCH FIRST 1 ROW ONLY
    - Db2 applies a sort avoidance preference with OPTIMIZE FOR 1 ROWS in access path selection
    - Sometimes avoiding a sort can result in a more expensive (in total cost) access path
      - OPTIMIZE FOR 2 ROWS can be specified to enable Db2 to consider use of a more efficient access path that uses a sort
      - Other integer values can be specified in OPTIMIZE FOR *n* ROWS, but OPTIMZE FOR 2 ROWS is recommended for this scenario

SELECT INTO….
OPTIMIZE FOR 2 ROWS

- Db2 13 FL 503
  - SELECT INTO statement support for OPTIMIZE FOR *n* ROWS
    - Can specify *optimize-clause* in SELECT INTO to enable Db2 to consider access paths using a sort
    - SELECT INTO statements always return a single row
      - OPTIMIZE FOR 2 ROWS can be specified to influence the optimizer

LISTAGG Function

- Db2 12 - FL501
  - New built-in aggregate function - LISTAGG
  - Provides a simple/efficient manner to collapse rows of column into a string
  - Used to aggregate a set of string values within a group into one string
  - An optional separator argument inserted between adjacent input strings
    - Can be used to delimit items in the result list
    - Specifying a comma as the separator produces a comma-separated list
  - Optional ordering can be specified for items in WITHIN GROUP clause
- Example
  - Produce an alphabetical list of comma-separated names, grouped by department, from EMPLOYEE table

```
SELECT WORKDEPT, LISTAGG(LASTNAME, ', ‘)
  WITHIN GROUP(ORDER BY LASTNAME) AS EMPLOYEES
 FROM EMPLOYEE GROUP BY WORKDEPT;
```

Application/SQL vs LISTAGG Performance - Example

- Prior to Db2 12
  - In order to provide a list of employees in each department
    - Define a cursor for SQL to query employee's name and department from EMPLOYEE table
    - Fetch data from cursor
    - Use application logic to evaluate whether employee's name is under same department
    - Perform concatenation
  - Requires users to implement the logic and concatenation
  - Can be error prone and uses time and resources
- Db2 12 – FL501
  - LISTAGG - Db2 built-in function can provide same function

```
SELECT DEPARTMENT, LISTAGG(ALL EMP_NAME, '; ')
WITHIN GROUP (ORDER BY EMP_NAME) AS EMP_NAME_LIST
FROM EMPLOYEE
GROUP BY DEPARTMENT;
```

DEPARTMENT	EMP_NAME_LIST
20	Alex; Susan
30	Emily; Catherine; Jim
40	Chris; John; Travis;

97% CPU reduction
94% elapsed time reduction

IBM Db2 12 Performance Redbook SG248404

SUBSTR Performance with LISTAGG Function

- Db2 13
  - When LISTAGG built-in function is used as the first argument for SUBSTR function
    - And both the start and length of the SUBSTR are literal values
    - Db2 sort is able to condense internal result buffer of LISTAGG function from the default of 4000 bytes to size of the SUBSTR setting

```
SELECT SUBSTR(LISTAGG(C1, ‘,’),1,100) FROM T1
```

  - Instead of allocating a sort result buffer size of 4000 bytes, Db2 allocates a buffer with a size of 100 bytes
  - Results in reduction of work file usage and improve the performance of SUBSTR function.

Hashing Function

- Db2 12 – FL501
  - Introduces hashing functions
    - One of many new built-in function
  - Hashing functions in a database are typically used to convert a character string into a fixed length value (or key)
    - Hash value is a representation of the original character string
  - Using hashed values can help with query performance
    - If hashed value is shorter in length than the original character string
  - Hashed values are often stored in the database and used as an index for data retrieval
- Set of hashing functions with varying degrees of cryptographic strength
- Typically stronger cryptographic hashes carry a longer execution time
- Db2 now provides several hashing functions

  User requirements differ, and Db2 provides a variety of functions

HASH_CRC32
HASH_MD5
HASH_SHA1
HASH_SHA256

Create/Replace for Procedures

- Db2 12 (FL507 – APAR PH24324)
    - Can better manage stored procedures in complex application environments
    - CREATE PROCEDURE statement (external and native SQL procedures) is extended to allow a new OR REPLACE clause
    - By adding ‘OR REPLACE’ clause to CREATE PROCEDURE statement
        - Can reuse original statement, make some changes to it, and reissue it to change the definition of an existing procedure
        - Eliminates need to first drop existing procedure and recreate it with original CREATE statement that has been modified as needed
    - Additionally, for native SQL procedures, can use ‘OR REPLACE’ clause on a CREATE PROCEDURE statement with a VERSION clause to replace an existing version of procedure, or to add a new version of the procedure
- When reusing a CREATE statement with the ‘OR REPLACE’ clause to replace an existing version or to add a new version of a native SQL procedure, result is similar to using an ALTER PROCEDURE statement with REPLACE VERSION or ADD VERSION clause
- If ‘OR REPLACE’ clause is specified on a CREATE statement and a procedure with specified name does not yet exist, the clause is ignored and a new procedure is still created

```
>>-----CREATE--------------------------procedure-name-----<<
                   |--OR REPLACE--|
```

SQL Data Insights

- SQL Data Insights (SQL DI) - Optional 13 Feature
  - Artificial intelligence (AI) functionality in the Db2 engine
    - Identifies valuable hidden insight within a relational database for better business decisions
    - Reduces the time necessary to take advantage of AI
    - No extract-transform-load (ETL) processes to move the data off mainframe
    - No need for machine-learning models to be designed by expert data scientist
  - SQL DI user interface can be added a Db2 table or view as an AI object
    - Enable AI query functionality
    - Run semantic AI queries directly against the Db2 table or view
      - Providing insight into the data and make business decisions
    - Three SQL DI Db2 built-in functions for semantic queries
      - Used in queries to infer hidden relationships between different entities in a table or view

- AI_SIMILARITY
  - Find similar or dissimilar entities in a table or view
- AI_SEMANTIC_CLUSTER
  - Check what other entities exist in table or view that could belong to a cluster of up to 3 entities
- AI_ANALOGY
  - Determines whether the relationship of a pair of entities applies to a second pair of entities

Summary

Db2 12+/13 Database/Application Design/Performance

- Db2 12 and 13 for z/OS offers many opportunities for improving designs, performance and increasing availability
  - Database Features
    - Multi-table migration to PBG
    - Partition insert with limit key
    - PBG to PBR online conversion
    - Selective Fast Traverse Block
    - Online LOAD Replace
    - z15 Huffman compression
  - SQL and Application Features
    - Rebind phase-in
    - SQL Data Insight
    - Lock granularity control
    - Thread storage improvements
    - LISTAGG and HASH
    - Improved Index-Lookaside
- Several new features are designed to provide better performance and availability
  - Must consider what it may take to implement and future maintainability
  - Must consider true usage capabilities

Db2 12+/13 – IBM References

- IBM Redbooks
  - Db2 12 for z/OS Technical Overview – SG248383
  - Db2 12 for z/OS Performance Topics - SG248404
  - Db2 13 for z/OS and More – SG248527
  - Db2 13 for z/OS Performance Topics - SG248536
- Db2 13 for z/OS Documentation

  https://www.ibm.com/docs/en/db2-for-zos/13
- World of Db2

  http://www.worldofdb2.com/

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

- **Db2 z/OS Courses**
  - Db2 12+/13 for z/OS Database and Application Performance Features
    - Application and DBA
  - Db2 12 for z/OS DBA Certification Crammer
    - Covers all exam topics
  - 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
  - Db2 Performance Data Collection and Exploitation

WWW.YLASSOC.COM

All classes are customized based upon customer requirements

- **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!