Achieving high availability, efficiency and application stability with Db2 13 for z/OS

June 19, 2022

Frances Villafuerte
Senior Software Engineer
francesv@us.ibm.com

IBM Db2 13 for z/OS

Available May 31, 2022

Agenda

- Review of Db2 strategic Universal Table Space
- Higher availability and performance for PBG table space in Db2 13
- Db2 table space conversion
- Application management
  - Application time out special register
  - Deadlock control - global variable
- References

Db2 Strategic Table Space – Universal Table Space

- Partition-by-Growth (PBG)
  - Designed to replace the classic segmented table space with ability to grow space by demand
  - No partition limit key
  - NPI for clustering index
    - Unsuitable for XXL size large tables because of performance and data management complexity
  - Strategy for utility operations can be a challenge
- Partition-by-Range (PBR)
  - Designed to replace the classic partitioned table space
  - Has all functionality of classic partitioned table space
- Partition-by-Range Relative page number (PBR RPN)
  - Provides better scalability and usability
    - Bigger table space, independent of partition size
    - Immediate ALTER of partition size
    - Lifts partition size limitation
  - Db2 future strategic range partition table space

V13 Enhancements of Partition by Range RPN

- System ZPARM
  - PAGE SET PAGE NUMBERING
    - ABSOLUTE for created range partitioned table spaces (PBR UTS)
    - RELATIVE for creating PBR RPN
  - Db2 13 the default value is changed to RELATIVE at V13R1M100
- Performance improvements in Db2 V13
  - Improved locking hash value to avoid false lock contention, specially for row level locking

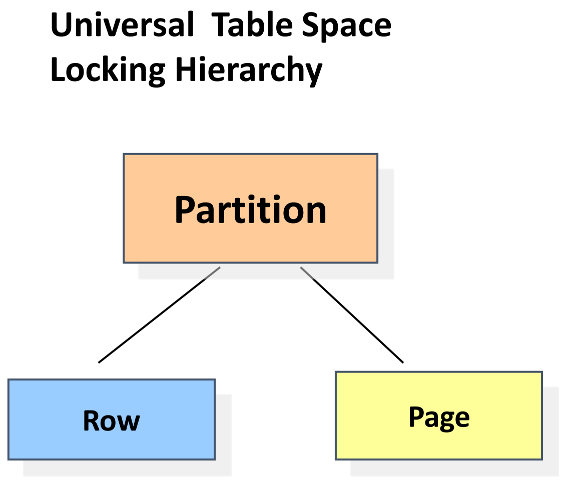
Partition by Growth (PBG)

Availability & Performance Improvements in **Db2 13**

Lesson learned from the common use of PBG

- Replacement of classic segmented table space
  - Table space can grow too large due to data volume
    - A database administrator has a great challenge to tune the concurrent insert application to execute optimally
  - Unsuitable for XXL size large tables because of performance and data management complexity
- Some of common concern:
  - PBG conditional partition lock failure causes application terminated prematurely
  - Insert application failed with incomprehensible message
    - SQLCODE -904 with Reason code 00C90090 (Conditional lock failure) or 00C9009C (part is full and PBG exceed max partition defined)
  - Unpredictable space reuse behavior from cross-partitioned search
  - Performance side effects from searching full partitions of a large table space

PBG locking scheme of crossed partition search during insert



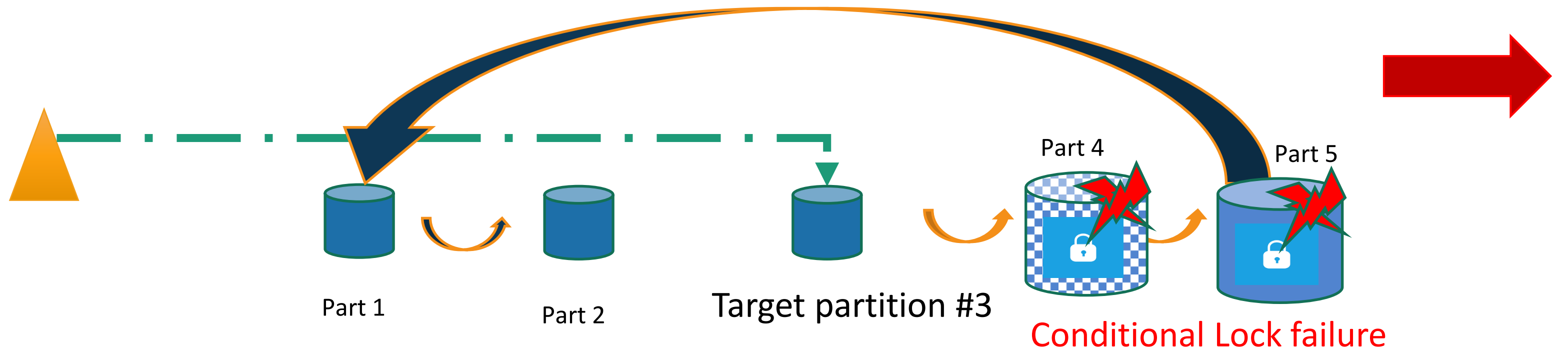
- Acquire partition lock conditionally
  - Followed locking hierarchy – partition lock is required to access partition
  - Reaching optimal performance for multiple PBG table space, partition lock is obtained conditionally
    - Unable to obtain conditional partition lock, the partition is skipped
    - Could result in resource not available after searched entire table space
  - After searching through all existing partitions once, Db2 fails INSERT transaction with -904 resource not available with reason code either 00C9009C(part is full and PBG exceed max partition defined) or 00C90090 (partition lock failure)
  - Conditional partition lock failure causes application terminated prematurely

For example

Searching partition in an ascending sequence by starting at partition 3 as determined by cluster index:

Assuming part 4 is not full but failed with conditional lock

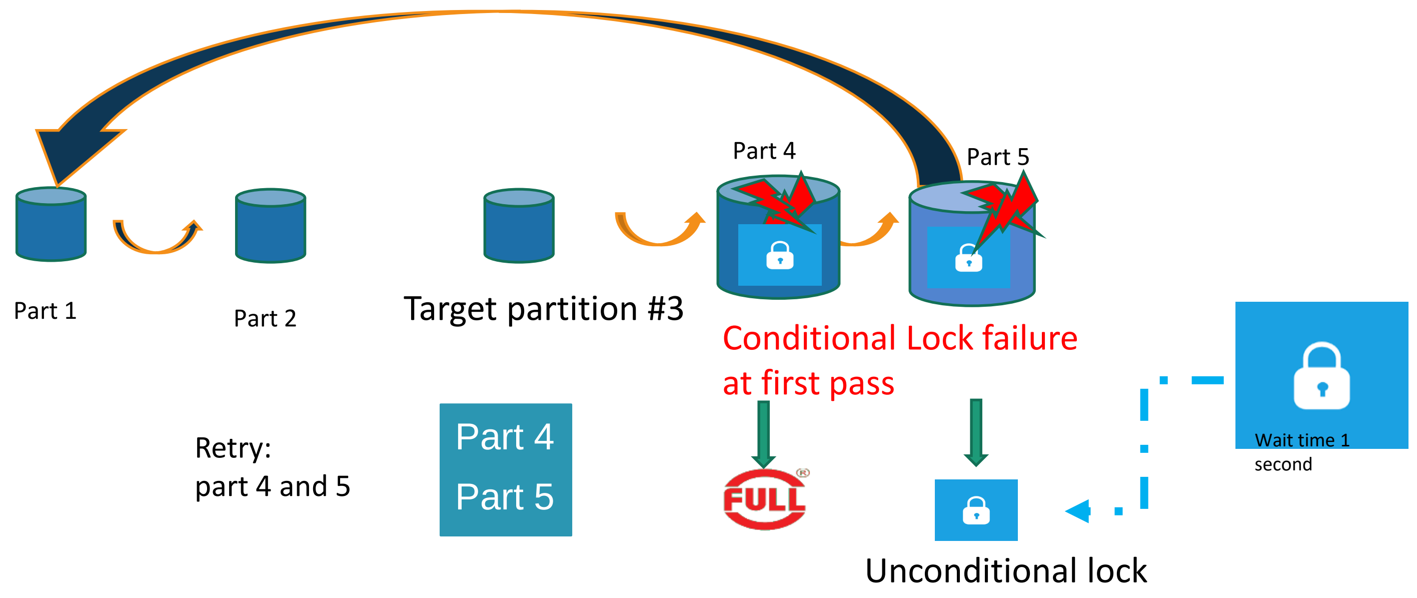
The partition searching sequence is 3->4->5->1->2 end all of searching



Insert failed and Issue SQLCODE -904 with reason code 00C9009C message
Because the last part searched is part 2 and it is full

Achieving high availability with PBG in Db2 13 for insert

- Retry partition with conditional partition lock failure previously
  - Retry up to 5 non-full partitions
  - Selectively retry partition with conditional or unconditional partition lock
    - If retry unconditionally, limit lock wait time to be approximal 1-2 second
    - If the unconditional retry failed, then SQL -911 with 00C9008E message will be issued which can help the user to find out the lock blocker and take corrective actions
  - If all parts are retried and full, allowing adding a new partition
  - Provides higher insert successful rate



```
DSNT376I -DB2A PLAN=DSNTEP3 WITH 691
        CORRELATION-ID=INSERTA4 CONNECTION-ID=BATCH
        LUW-ID=DSNCAT.SYEC1DB2.DAE53CAF57D2=9
        ……

        ONE HOLDER OF THE RESOURCE IS PLAN=DSNTEP3
        WITH CORRELATION-ID=TQI01005 CONNECTION-
        ID=BATCH
        …..
        :*:* ON MEMBER DB2A
        REQUESTER USING TIMEOUT VALUE=1 FROM
        IRLMRWT
        HOLDER USING TIMEOUT VALUE=60 FROM IRLMRWT
```

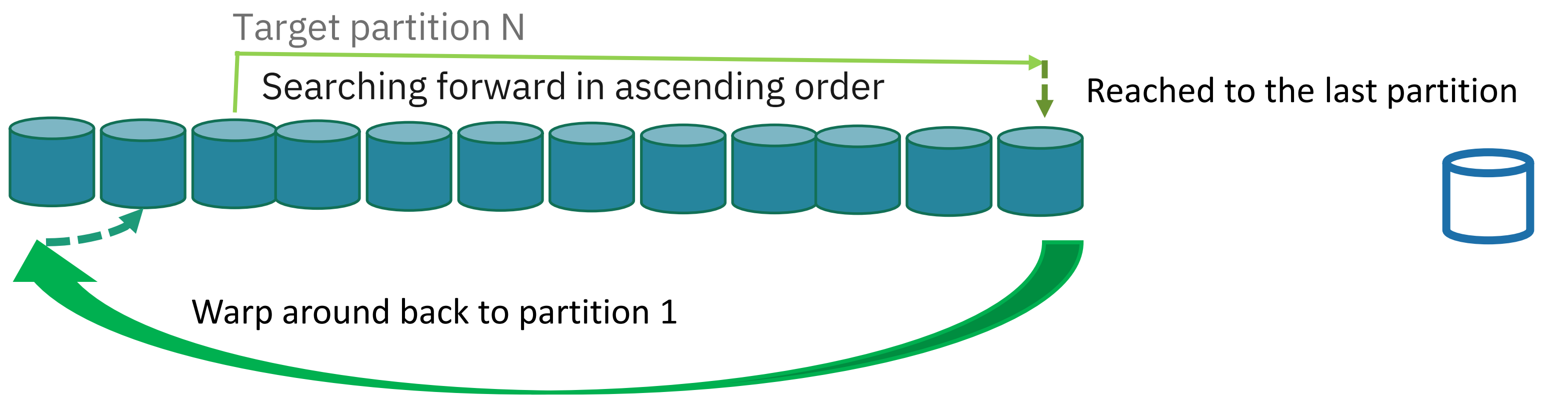
Db2 13 - IFCID 02 – Statistic Data

- New counters added to IFCID 02 Data Manager Data during the retry process

  - **CONDITIONAL LOCK FAILURE** (QISTCONDLKF)
    Number of failed conditional lock request during Insert operation

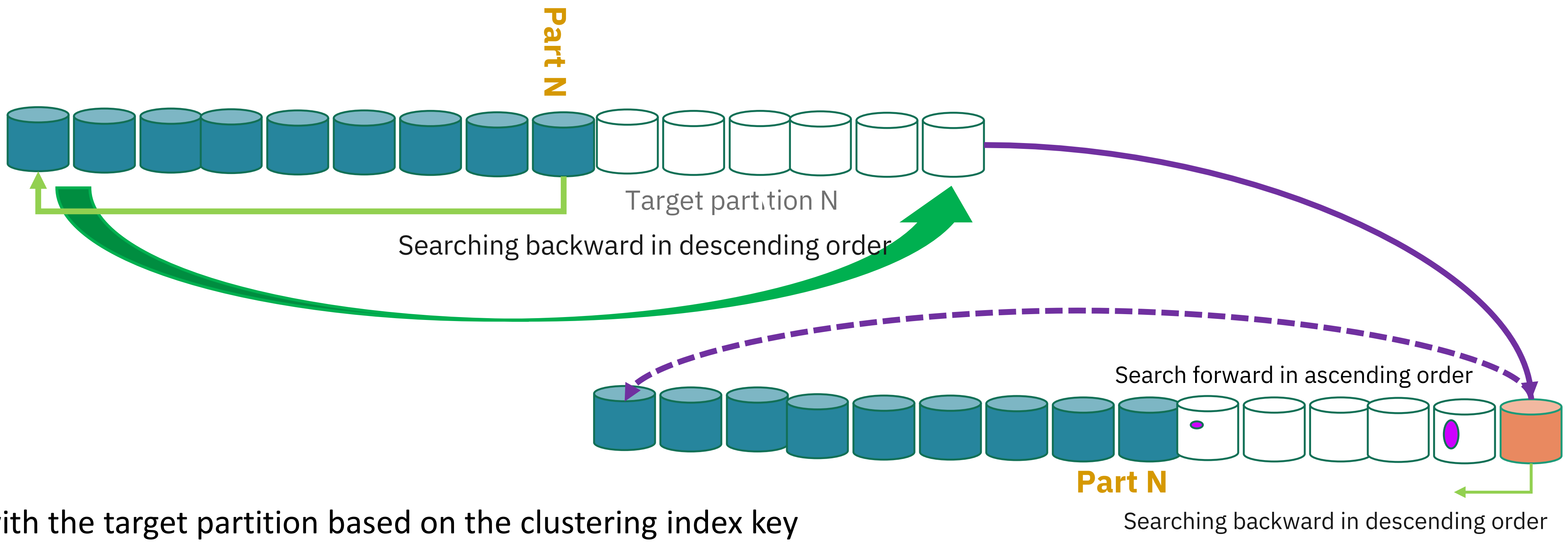
  - **UNCONDITIONAL LOCK RETRIES** (QISTRETRYLK)
    Number of times a failed conditional lock request has been retried with an unconditional

PBG crossed partition search algorithm – Ascending sequence


Target partition N
Searching forward in ascending order
Reached to the last partition
Warp around back to partition 1

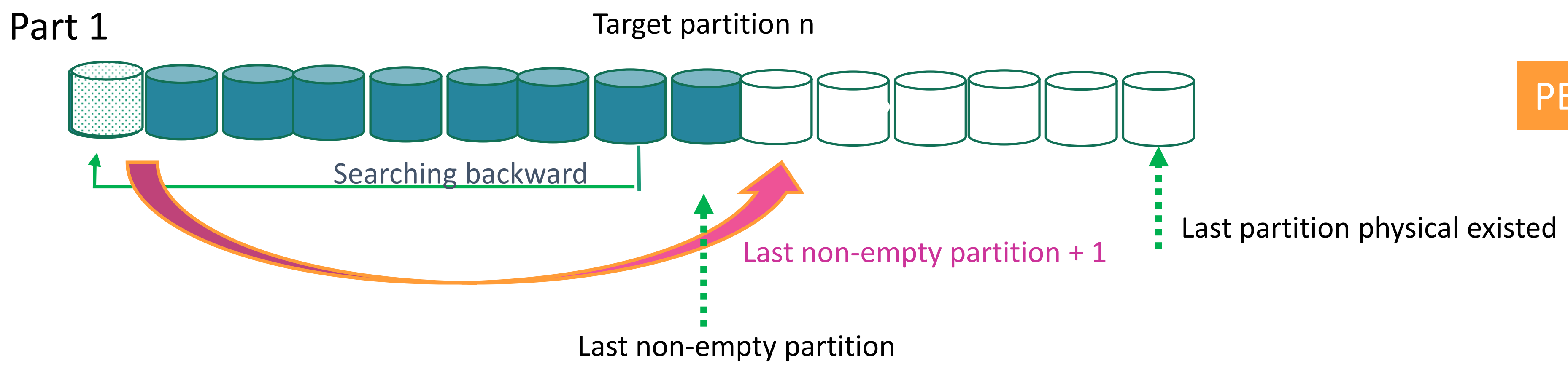
- Start with the target partition first, the one based on the clustering index key
- Searching forward in the ascending partition order
  - Starting at the target partition N, then N+1, N+2 ... Etc
  - When the last partition is full, then goes to the partition 1 and search forward to the target partition N
  - Adds a new partition
- Achieving higher performance with Db2 13
  - Optimization is in place to prevent every thread searching through full partitions
    - Full partition is tracked at each data sharing member
  - High % of performance improvement observed specially for random inserts
  - Small amount of the delete spread around the table space still requires searching of each partition
  - Workload balance among insert/delete operation is essential to take advantage of this improvement

PBG crossed partition search algorithm – Descending sequence prior to Db2 13


Part N
Target partition N
Searching backward in descending order
Search forward in ascending order
Part N
Searching backward in descending order

- Start with the target partition based on the clustering index key
- Searching in a descending partition order
  - Starting at the target partition N, then N-1, N-2 ... Etc
  - When reaches to partition 1, then goes to the last partition of table space
- Side effects of descending crossed-partition search
  - It is designed in a way that is beneficial for some niche use cases but creates side effects for others
  - Space is hard to manage
  - Embedded empty partition requires REORG on entire TS in order to be deleted

Db2 13 Improvement of PBG Descending crossed-partition Search



PBG table space only

When descending partition search is used:

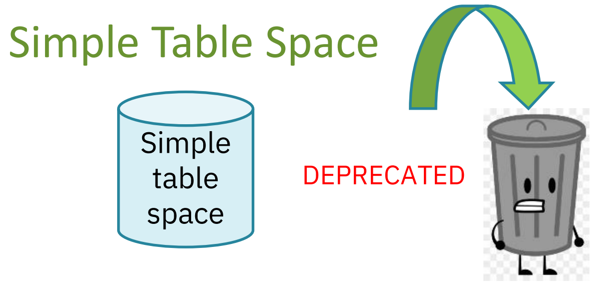
- Search in sequential descending order as before
- When partition 1 is reached, Utilizes RTS information to determine the next searching partition
  - Internally evaluates free space from RTS between tracked non-empty partition and last physical partition
  - In doing so, partitions in between the original partition and the destination partition will be skipped
- Reduce possibility of embedded empty partitions between last defined partition and the last non-empty partition
  - For the case where Db2 does not have last non-empty partition tracked after physical open, and the target search starts with partition 1, the next target partition will be the last defined partition.
- Reduce required of REORG to reclaim empty partitions

Ready to convert table spaces with Pending Alter ?

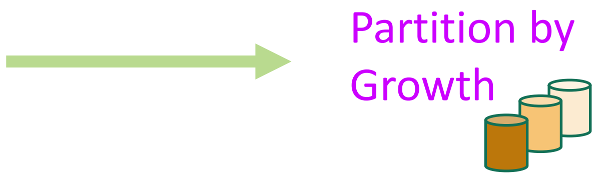
Pending alter:

ALTER statement follow by REORG to materialize the structure change

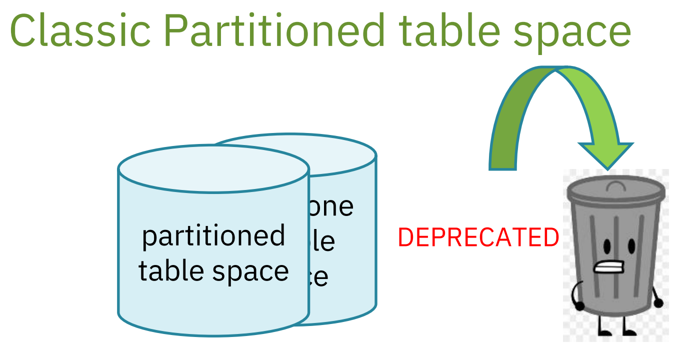
Object Conversion Evolution (1 of 5)



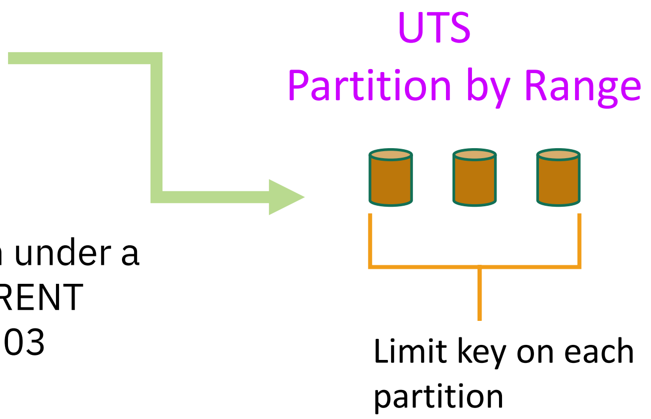
ALTER TABLESPACE ...**MAXPARTITIONS** n



- **If single table in the table space can be converted to PBG**
- Continue to use but can not created a new one
- Deprecated in **V9**



ALTER TABLESPACE ...**SEGSIZE n**



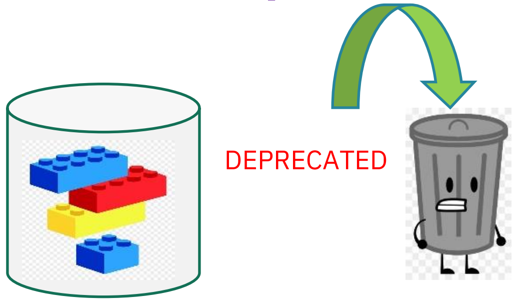
- Requires table controlled partitioned limit key
- Deprecated, but can continue to use existing one
  - In **V12 M504** or later with any DDL when run under a package with APPLCOMPAT <= 503 or CURRENT APPLICATION COMPATIBILITY register <=503

ALTER TABLESPACE ...**PAGENUM RELATIVE**

- Converted to PBR UTS -> PBR RPN
- Available in V12 M500

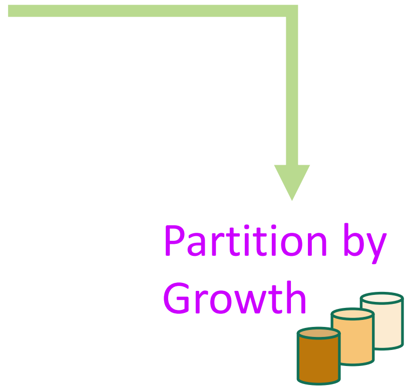
Object Conversion Evolution (2 of 5)

Segmented table space



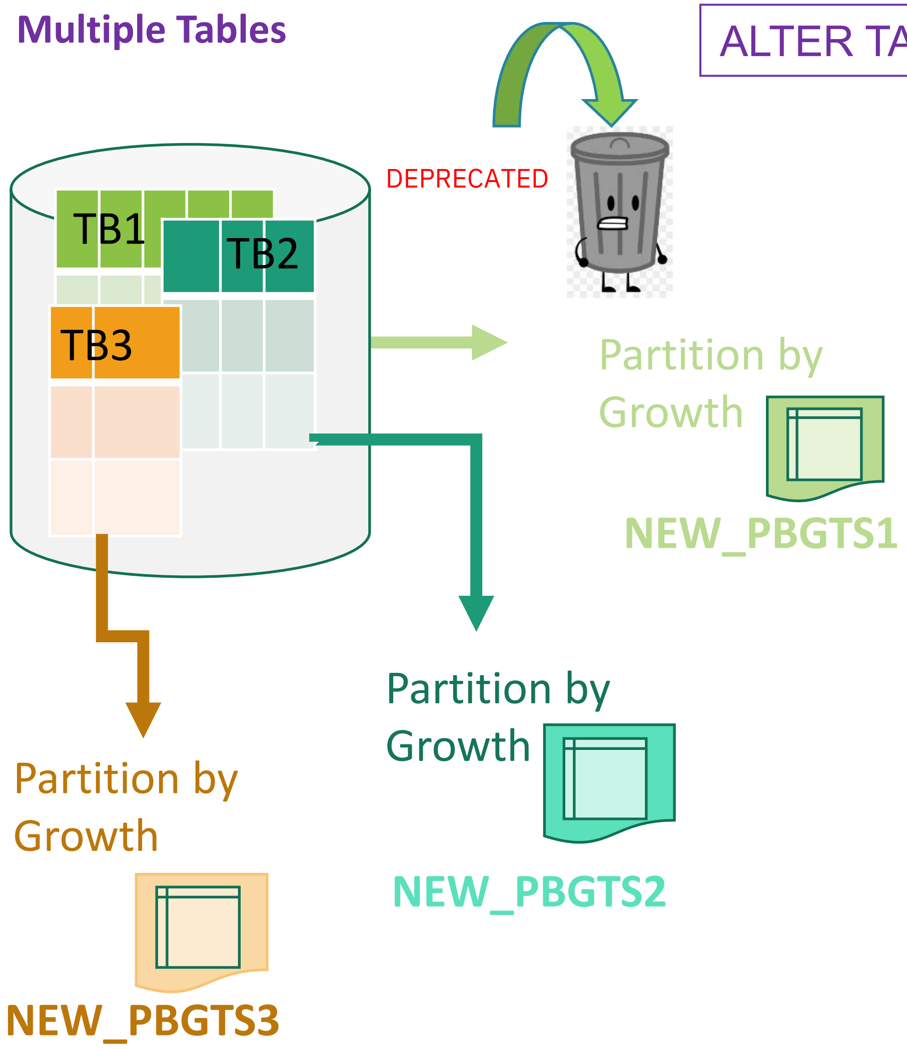
ALTER TABLESPACE …**MAXPARTITIONS** n

Single table in the table space



- A single table in the table space can be converted to **PBG,**
- Pending ALTER followed by a REORG
- Deprecated, but can continue to use existing one
  - In **V12 M504** or later
  - DDL to create segmented table space can run under a package with APPLCOMPAT <= 503 or CURRENT APPLICATION COMPATIBILITY register <=503

Object Conversion Evolution (3 of 5)



ALTER TABLESPACE …**MOVE TABLE** Tbname **TO TABLESPACE** newtsname

- Available in **Db2 12 FL508**
- To convert a multi-table table space to individual PBG UTSs, following are the steps:
  1. For each table in the table space:
     - Create the target PBG UTS
     - PBG UTS in the same database with DEFINE NO,MAXPARTITIONS 1
     - The target table space must be created prior to executing ALTER TABLESPACE MOVE TABLE statement
  2. Execute an ALTER statement as a pending definition change to move the table to the target PBG UTS
     - One table per ALTER TABLESPACE statement
  3. Run REORG TABLESPACE against the multi-table table space to materialize the pending ALTER statements
     - Multiple pending ALTERs can be stacked and materialized in a single REORG of the source table space
  4. Possibly rebind invalidated packages
     - Invalidation of packages and cached dynamic statements dependent on the tables moved (not the entire table space), and inline statistics are not collected.

Object Conversion Evolution (4 of 5) – Example of MOVE TABLE

- Example of a multi-table table space DB1.TS1
  - Containing 3 tables (TB1, TB2, TB3),
    - All 3 tables are moved out into individual target PBG UTSs (DB1.NEW_PBGTS1, DB1.NEW_PBGTS2, DB1.NEW_PBGTS3),
    - The source table space DB1.TS1 can be dropped after REORG

  - Create the new target PBG UTSs with DEFINE NO and the desired table space attributes:

```
CREATE TABLESPACE NEW_PBGTS1 IN DB1 MAXPARTITIONS 1 DEFINE NO;
CREATE TABLESPACE NEW_PBGTS2 IN DB1 MAXPARTITIONS 1 DEFINE NO;
CREATE TABLESPACE NEW_PBGTS3 IN DB1 MAXPARTITIONS 1 DEFINE NO;
```

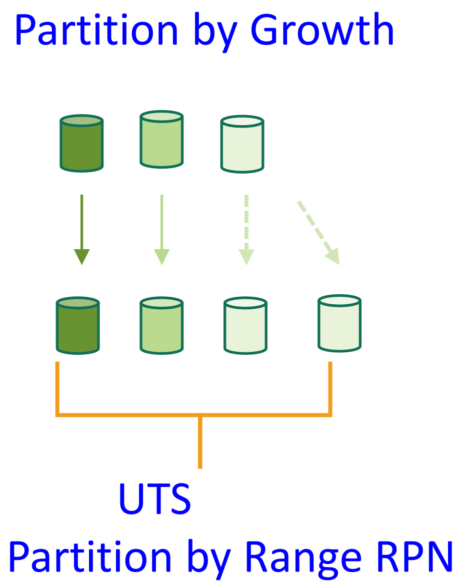
  - Execute ALTER TABLESPACE statements to move the tables, after which the ALTER statements are saved away in SYSPENDINGDDL as pending definition changes.

```
ALTER TABLESPACE DB1.TS1 MOVE TABLE TB1 TO TABLESPACE DB1.NEW_PBGTS1;
ALTER TABLESPACE DB1.TS1 MOVE TABLE TB2 TO TABLESPACE DB1.NEW_PBGTS2;
ALTER TABLESPACE DB1.TS1 MOVE TABLE TB3 TO TABLESPACE DB1.NEW_PBGTS3;
```

Object Conversion Evolution (5 of 6) PBG to PBR

- Achieving better availability with Online conversion from PBG to PBR
  - Next major step in online schema evolution strategy to convert different table space type in UTS
  - Addresses increasing size of PBG table spaces that are better suited with partitioned by range
  - Building on pending alter functionality with new ALTER syntax
  - Default to PBR with RELATIVE PAGE NUMBERING table space

ALTER TABLE….**ALTER PARTITIONING TO PARTITION BY …**



```
ALTER TABLE SCR001.TB01 ALTER PARTITIONING TO
                         PARTITION BY RANGE (ACCT_NUM)
                        ( PARTITION 1 ENDING AT (199),
                         PARTITION 2 ENDING AT (299),
                         PARTITION 3 ENDING AT (399),
                         PARTITION 4 ENDING AT (MAXVALUE));
```

Converting PBG to PBR Considerations (1 of 4)

- Pending or Immediate alteration
  - If the physical data set is defined and allocated – Pending Alter
  - If the physical data set is not defined – Immediate Alter
- REORG materialized pending alter
  - REORG TABLESPACE with SHRLEVEL REFERENCE or CHANGE of the entire table space.
- The existing DDL_MATERIALIZATION subsystem
  - Does not apply to this conversion
- Converted partition-by-range (PBR) table space attributes:
  - The number of partitions can be difference between converted PBR and the original PBG table space
  - The newly added partition(s) inherit(s) most attributes from the table space level.
  - There are no changes to the table space OBIDs and table OBID
  - The converted PBR table space will use **relative page numbering (RPN)**
  - The PBR RPN requires EA/EF datasets (extended addressability / extended format).
  - Partitioned range uses Table Control Partitioning scheme

Converting PBG to PBR Considerations (2 of 4)

Index management:

- All indexes defined on a table in partition-by-growth (PBG) are non-partitioned indexes (NPI).
- The existing NPI indexes on the table are handled as part of the conversion. Db2 will not change any aspects or attributes of these indexes.
- Partitioned indexes (PI) will not be created during this conversion process. If desired, they can be created after the conversion has been materialized.

Access path consideration:

- After the PBG to PBR conversion, the access path in place for queries involving the table might now qualify for page range access (PAGE_RANGE = Y in the PLAN_TABLE)
  - Any BIND or REBIND that is done with APREUSE or APCOMPARE will not flag that access path change as an error.
  - The change will be tolerated so that the BIND or REBIND can complete successfully.

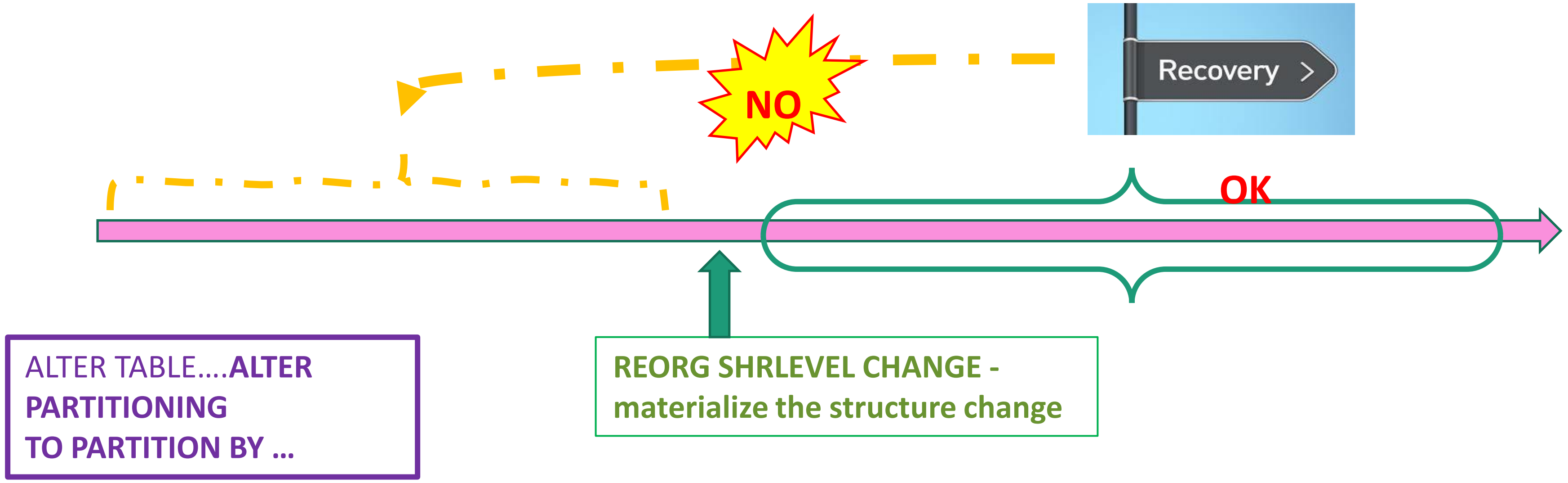
Converting PBG to PBR Considerations (4 of 4)

- The table must not be any of the following
  - A catalog or directory table
  - Created global temporary table
  - A Hash Table space
  - Involved in a clone relationship
- The table must not contain:
  - LOB column(s)
  - XML column(s)
- The high limit key for the last partition
  - Ascending Key - The last partition requires MAXVALUE
  - Descending Key – The last partition requires MINVALUE
- The PBG table space cannot have existing pending alter
- The PBG table space cannot be in any restricted state

```
ALTER TABLE SCR001.TB01 ALTER PARTITIONING TO
                        PARTITION BY RANGE (ACCT_NUM)
                       ( PARTITION 1 ENDING AT (199),
                        PARTITION 2 ENDING AT (299),
                        PARTITION 3 ENDING AT (399),
                        PARTITION 4 ENDING AT (MAXVALUE));
```

Recovery Consideration

- Recovery after converting PBG to PBR RPN
  - After a REORG has materialized a pending change of object conversion RECOVER utility will
    - support recovery to the current state or to a point in time after this materializing REORG
    - prohibit recovery to a point in time before this REORG.


Recovery
NO
OK
ALTER TABLE....ALTER PARTITIONING TO PARTITION BY ...
REORG SHRLEVEL CHANGE - materialize the structure change

Application Managements

Locking control of Application – Problem Statements

- Difficult to manage locking serialization among applications
  - Different applications have different characteristics but have no way to express their own priorities and wait time in case of lock contention.
  - Possible application by-pass may be to split off into its own Db2 subsystem to provide for application affinities or redesign application. However, this increases management cost and overhead, and decreases availability and resiliency.
- The subsystem Parameter (IRLMRWT)
  - Timeout wait time before IRLM reject the lock.
  - Governing all the applications running in the subsystem
  - Subsystem level adjustments, does not provide enough granularity for each individual application needs
- Certain DDL activities can be prone to deadlocks
  - If the thread performing the DDL is chosen as the victim then scheduled DDL activities may fail, requiring repeat attempts and/or impacting subsequent work.
  - The resolution of a deadlock and choosing the deadlock victim are based on Db2 internal evaluation factors.

Application Lock Timeout Control

- New special register - CURRENT LOCK TIMEOUT
  - The data type of the register is INTEGER, and the valid value is between -1 and 32767, inclusive, or the null value
    - **NULL:** Default uses IRLMRWT value
    - **WAIT (-1)**: No time out
    - **NOT WAIT (0)**: Application is not waiting for the lock
  - Provides higher availability and Usability
  - Gives granularity of lock time out control at the application level
  - It overrides the IRLMRWT subsystem parameter and controls the number of seconds to wait before a resource timeout is detected.
  - IRLMRWT subsystem parameter is online changeable in 13
  - Application profile control also improved to support this new special register
  - New SPREG_LOCK_TIMEOUT_MAX parameter, this adds a new field, LOCK TIMEOUT MAX to installation panel.
    - Controls the maximum value that can be specified in a SET CURRENT LOCK TIMEOUT statement
  - Available at V13 FL 500 (V13R1M500) or higher
  - IRLM 2.3 PTF

Application with deadlock control

- New global variable DEADLOCK_RESOLUTION_PRIORITY
  - Influence deadlock resolution at application level through the new global variable
  - Provides high application availability and usability
  - This built-in global variable specifies a relative priority to be used in resolving deadlocks with other threads
    - The higher value of the global variable, the higher priority to get the resource
  - The data type of the built-in global variable is SMALLINT, and the valid value is between 0 and 255, inclusive, or the null value.
  - Application profile control also improved to support this new global variable
  - Available at V13 FL 500 (V13R1M500) or higher
  - IRLM 2.3 PTF

Better Application Usability with Improved Message

DSNT376I PLAN=plan-name1 WITH CORRELATION-ID=correlation-id1 CONNECTION-ID=connection-id1 LUW-ID=luw-id1 THREAD-INFO=thread- information1 IS **TIMED OUT**.

ONE HOLDER OF THE RESOURCE IS

PLAN=plan-name2 WITH CORRELATION-ID=correlation-id2 CONNECTION-ID=connection-id2 LUW-ID=luw-id2 THREAD-INFO= thread-information2 ON MEMBER member-name.

REQUESTER USING TIMEOUT VALUE <tout-interval1> FROM <timeout-source1>.

HOLDER USING TIMEOUT VALUE <tout-interval2> from <timeout-source2>.

- ***tout-interval1*** and ***tout-interval2*** *wait interval in seconds before a lock request timed out*
- ***timeout-source1*** and ***timeout-source2*** the timeout interval specification that was used for the timeout, such as IRLMRWT or special register

```
DSNT376I -DB2A PLAN=DSNTEP3 WITH 691
    CORRELATION-ID=INSERTA4 CONNECTION-ID=BATCH
    LUW-ID=DSNCAT.SYEC1DB2.DAE53CAF57D2=9
    ……

    ONE HOLDER OF THE RESOURCE IS PLAN=DSNTEP3 WITH CORRELATION-ID=TQI01005 CONNECTION-ID=BATCH
    …..
    :*:* ON MEMBER DB2A
    REQUESTER USING TIMEOUT VALUE=10 FROM Special Register
    HOLDER USING TIMEOUT VALUE=60 FROM IRLMRWT
```

Better Availability and Usability in Application management

- some of IFCID improvements
  - IFCID437 is a new trace record for execution of the SET CURRENT LOCK TIMEOUT statement
  - IFCID 2 and 3**:** new counters are introduced for accounting and statistics traces
  - IFCID 106: is modified to trace the new SPREG_LOCK_TIMEOUT_MAX subsystem parameter
  - IFCID 172: A new flag is added to indicate the deadlock value comes from global variable or the Db2 internal factors.

Summary

- Better performance for PBR RPN (Partitioned-by-Range Relative page numbering)
  - Locking hash algorithm changes to provide better performance specially for row level locking
  - It is the default partitioned table space
- PBG improvements
  - Availability enhancements in crossed partition search
  - Improved performance
- Object conversion case
  - Allow to convert PBG -> PBR RPN
- Application management control
  - Lock timeout control at application level with special register - CURRENT LOCK TIMEOUT
  - DeadLock victim control at application level with global variable - DEADLOCK_RESOLUTION_PRIORITY
  - Profile supports both – new special register and global variable

References

- Db2 Redbook – IBM Db2 13 for z/OS and more
  https://www.redbooks.ibm.com/abstracts/sg248527.html

- Db2 13 for z/OS
  https://www.ibm.com/docs/en/db2-for-zos/13

Thank You

Speaker Name : Frances Villafuerte

Senior Software Engineer

Db2 Core Engine development

francesv@us.ibm.com