IDUG

2026

Sydney | March 16 - 18

AU Db2 TECH CONFERENCE

Improved Availability through the Evolution of Universal Table Space (UTS)

Anthony Ciabattoni, IBM

Db2 for z/OS SWAT Team

Session Code: A08

Platform:

Db2 for z/OS

Agenda

- Table Space History
- Universal Table Spaces
- Universal Table Space Conversions
- Db2 REBIND Access Path Stability
- Statement Level Invalidation
- Questions

History of Db2 base table types

Simplicity

- **Simplicity**
  - Simple table space
  - Easy to use
  - Capacity limitation
  - Data management challenge for multiple tables

Scalability

- **Scalability**
  - Partitioned table space - Larger object and be able to manage data in a selective subsets
  - More flexible for data volume growth in the future
  - Less flexible in data manipulation with larger data set

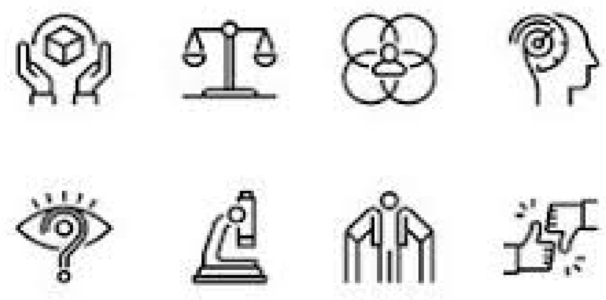
Better Data Manipulation structure

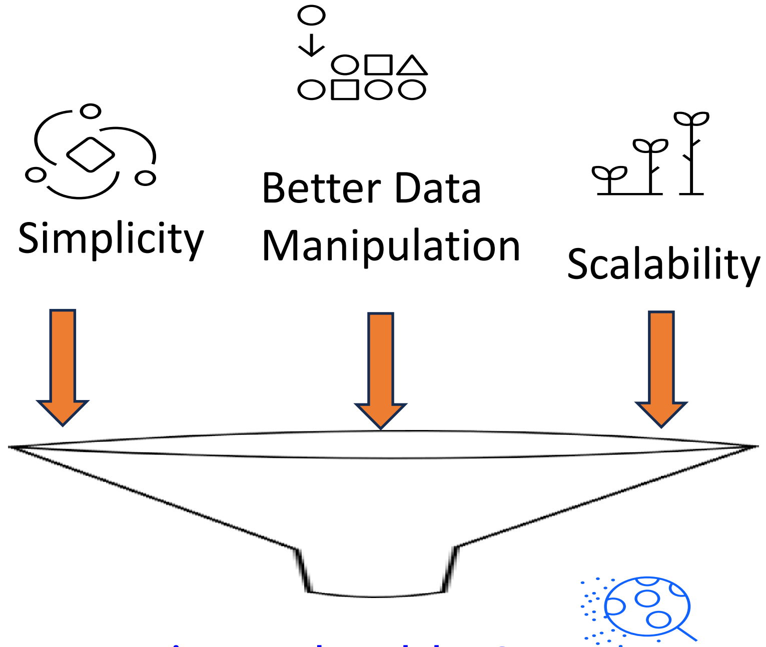
- **More flexible with data management**
  - Segmented table space
  - Higher data availability and data accessing performance
  - More flexible with multiple tables
  - Capacity limitation of 64G per table space

Why Universal Table Space (UTS)

- Why Universal Tablespaces ?
  - Bring together the advantage of partitioned and segmented table space
  - Partition scheme provides higher scalability
  - Segmented structure provides more flexible of data management
  - Single table per table space for simplicity
  - More Flexibility
  - New features/functionality

More Flexibilities





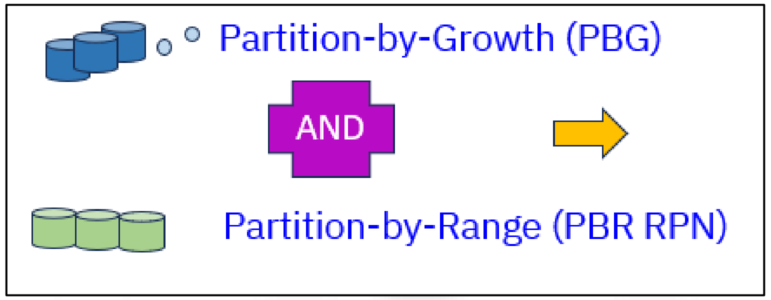
Universal Table Space

Why Universal Table Space (UTS) ...

- Why Universal Tablespaces ? ...
  - Support of non-UTS table spaces is approaching the end
  - All table spaces must be converted to Universal Table Spaces before migrating to the next release of Db2 (Db2 13+1)
    - Must develop a plan to convert any residual legacy table spaces to Universal Table Space

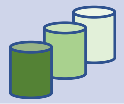
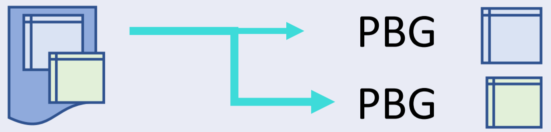
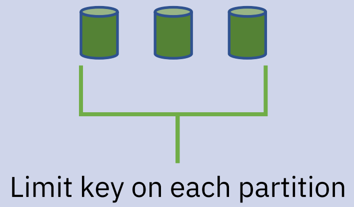
UTS general design rationale

- Partition by Growth (PBG) UTS
  - Should be seen as a replacement for classic table spaces
- Partition by Range (PBR) UTS
  - Should be seen as a replacement for classic partition table space
  - PBR UTS – Partition by Range Absolute page number
  - PBR UTS – Relative Page Number (RPN)
    - Strategic Direction
    - Partition level DSSIZE
    - Supersized UTS PBR
    - Up to 1TB partition size or 4 Petabytes of table space
    - Max number of rows of 4K/35 trillion rows for 32K pages



The strategic table space types for Db2 tables

Non-UTS to UTS table space conversion

Table space type	Pending ALTER DDL	Target converted table space type
Single table : Segmented / simple table space	ALTER TABLESPACE …**MAXPARTITIONS** n	Partitioned-by-growth (PBG UTS) 
Multiple tables: segmented	ALTER TABLESPACE …**MOVE TABLE** Tbname **TO TABLESPACE** TSname	
Classic partitioned table space	ALTER TABLESPACE …**SEGSIZE** n	Partitioned-by-Range (PBR UTS) 

Converting legacy table spaces

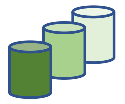
Simple Table Space or Segmented table space

A single table in the table space

ALTER TABLESPACE ...**MAXPARTITIONS** n

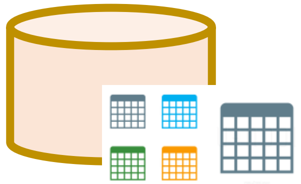
REORG table space

Partition by Growth



Segmented table space

Multiple Tables



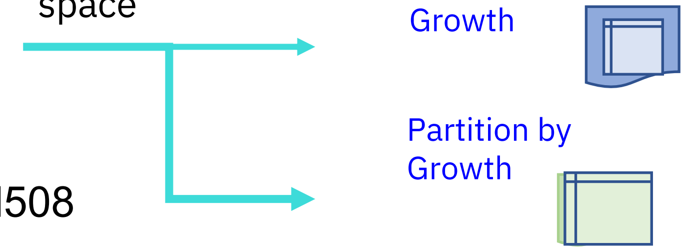
ALTER TABLESPACE ...**MOVE TABLE** Tbname

TO TABLESPACE newtsname

REORG table space

Partition by Growth

Partition by Growth



Db2 V12R1M508

- Non-UTS table space has been deprecated through out the releases
- Non-Universal table spaces are still to be supported in Db2 13
  - Simple table space is not supported since release 9
  - All other objects are initially deprecated @V12R1M504
- Converting multiple tables in segmented table space to PBG
  - Using MOVE TABLE DDL
  - Available in V12RM508

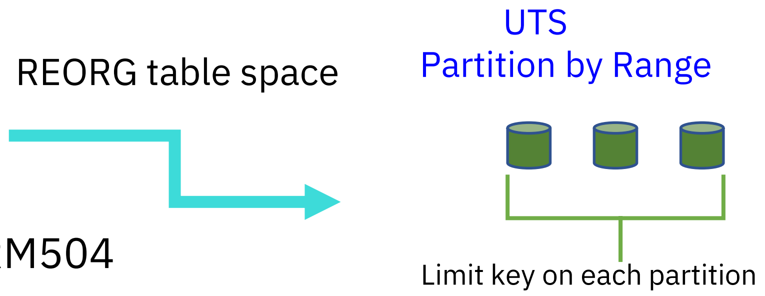
Converting classic partition table space

Classic Partitioned table space

ALTER TABLESPACE ...**SEGSIZE n**

REORG table space

Db2 V12RM504



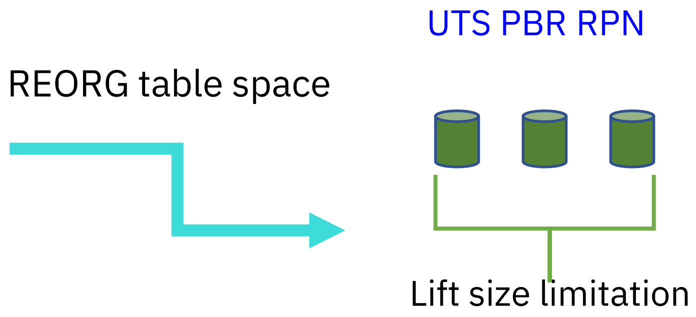
Classic Partitioned table space

ALTER TABLESPACE ...**SEGSIZE n**

Db2 V12RM504

ALTER TABLESPACE ...**PAGENUM RELATIVE**

REORG table space



- Converting classic partition table space to UTS PBR or UTS PBR RPN
  - Stacking two alters with one REORG materialize all the changes

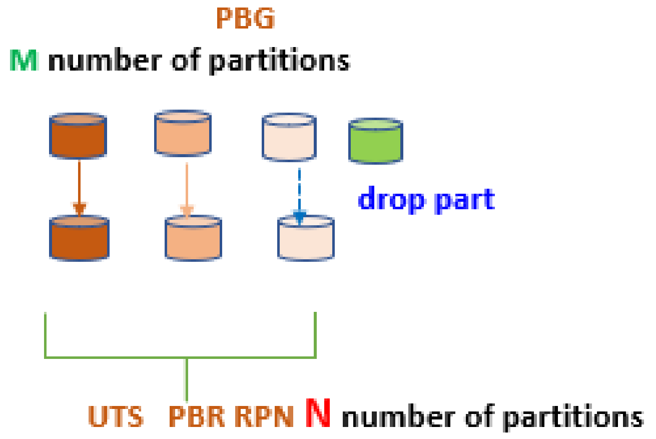
When to consider converting UTS PBG to UTS PBR RPN

- Under certain circumstances PBG UTS may no longer the optimal UTS choice
  - Table usage and data volume usage has changed over time
  - Incorrect historical decision based on the point-in-time knowledge
    - *“Set it and forget it”*
- Consequences
  - Insert or query performance degradation due to size of table space
    - Performance degradation can occur for large tables in PBG tables spaces, the size of the table space is often a major cause.
    - Db2 13 also introduces SQL INSERT enhancements for crossed partition search that can help with certain large PBG table space situation
    - Size of LOB table space associated with base table
      - ✓ PBR table space is more flexible to control the size of the LOB table space
  - Problems associated with very large non-partitioned indexes (NPI)
  - Difficulty completing REORG to maintain data clustering
  - Lack of parallelism features support for utilities
  - Limited support for partition-level utility operations
    - Minimal utility parallelism capabilities are fully supported for PBG table spaces
- Blog in IBM Data Management Community

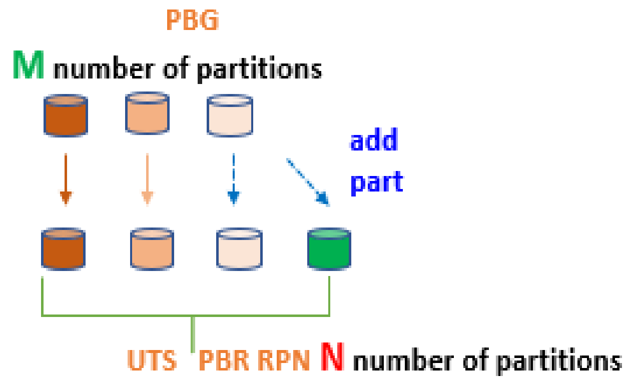
  https://community.ibm.com/community/user/blogs/frances-villafuerte/2022/08/15/convert-tables-in-pbg-table-spaces-to-pbr-db2-13?hlmlt=BL

Basic concept of converting PBG to PBR RPN

- There are no changes to the table space OBIDs and table OBID
- The converted table space is PBR RPN
- The number of partitions between two table spaces can be different
  - **For Replication** – the target converted table space must contain equal or more partitions
- The existing NPI indexes on the PBG table is still NPI
  - Db2 will not change any aspects or attributes of these indexes through conversion
- No Partitioned indexes (PI) will be created
  - If desired, user can create after the conversion has been materialized
- Conversion from PBG UTS to PBR UTS needs to meet the following conditions
  - High limit key for last partition requires MAXVALUE or MINVALUE
  - Cannot be an accelerated table
  - Cannot have unmaterialized pending changes



If *M* > *N* Db2 will drop extra partitions

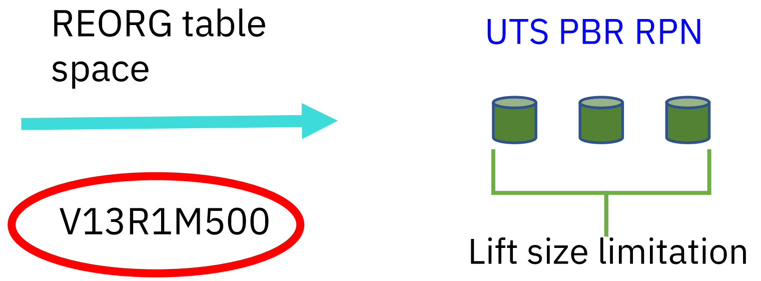


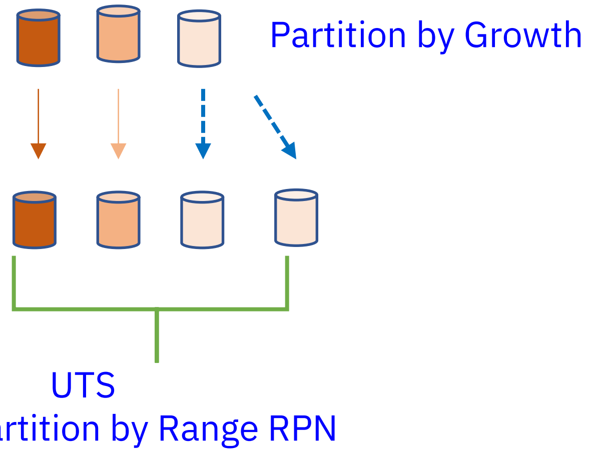
If M < *N*, Db2 will create the extra partitions for the converted PBR table space.

Converting from UTS PBG to UTS PBR RPN

UTS Partition by Growth (PBG)

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





```
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));
```

Partitioning key considerations PBG to RBR RPN

- Options of selecting partition key column
  - Exiting column from the table
    - No application changes required
    - No natural partitioning key for data distribution
  - Adding a new column to the table
    - Applications required
  - Utilizing Db2 managed value
    - Benefit
      - ✓ Db2 automatically manages a unique value, either in a monotonically ascending order within the table or randomized value such as ***ROWID*** column
      - ✓ Application does not need to manage the value of the key
  - ***ROWID***
    - Data type can uniquely identify rows in a Db2 subsystem
    - Generated value for the column when a row is inserted/REORG
    - Can be created as a ***hidden column***
      - Supported on CREATE TABLE starting Db2 12
    - Can be ALTER added to existing table
      - Enabled in APPLCOMPAT ***V13R1M506*** (UTS online)

Using Db2 managed value – ROWID column

- Characteristic of column with ROWID data type (aka ROWID column)
  - A data type can uniquely identify rows in a Db2 data sharing group
- For table has LOB column, the ROWID column is required
  - Column with ROWID data type can be created without LOB column
- ROWID value
  - The value is generated for the column when a row is inserted/REORG
  - If alter added,
    - Can specify Db2 to generated default value for existing rows during query
- It can be implicitly or explicitly created
  - Can be created as *hidden* column
    - It is supported on CREATE TABLE starting V12 NFM
  - Can be ALTER added to the existing table
    - *Enabled in APPLCOMPAT V13R1506*
    - Only supports on UTS table space
      - ✓ SQLCODE -270 for Non-UTS

Using Db2 managed value – ROWID column …

- Benefit
  - Creates as ***hidden***, no application changes
  - Unique value for each data row within the object
    - Value is generated from the scramble of LOG SEQUENCE number
    - Using for partition key for ease control of data distribution among table space
  - Once the value is generated it will not be changed at all
  - Easier to manage
- Alter added a ROWID column
  - No immediate REORG required to materialize the value of the column

Using Db2 managed value – ROWID column ...

- Example of ROWID value
  - Scramble 17 bytes of LOG sequence number
  - Internally format

A3E414F191F7B108609C014C6B20010000

A54AC14F191F7B609609C014C6B2001000

  - If ALTER add to the table,
    - Db2 generated the internal value base on the position of the row within the data base

644000008000011700030201000000000

Example of adding hidden ROWID column

```
CREATE  DATABASE TestDB   .....
 commit ;
 CREATE TABLESPACE TestTS IN TestDB  ....
              MAXPARTITIONS 5;
  commit ;
```

```
CREATE TABLE  TestTB

                 (COL1  BIGINT    DEFAULT 1,

                  ...

                  COL18 CHAR(160) DEFAULT 'COL18')

          IN TESTTS.TESTDB;
```

```
ALTER TABLE   TestTS.TestDB

        ADD COLUMN  HIDROWID ROWID

              NOT NULL GENERATED ALWAYS  IMPLICITLY HIDDEN;

COMMIT;
```

Example of using ROWID column as partitioning key

```
ALTER TABLE TestTS.TestTB ;

Commit ;

ALTER PARTITIONING TO PARTITION BY RANGE
 (HIDROWID)
 (PARTITION 1 ENDING AT (x'3000'),
  PARTITION 2 ENDING AT (x'6000'),
  PARTITION 3 ENDING AT (x'9000'),
  PARTITION 4 ENDING AT (x'C000'),
  PARTITION 5 ENDING AT (x'D000'),
  PARTITION 6 ENDING AT (x'F000'),
  PARTITION 7 ENDING AT (MAXVALUE)
 )
;
```

Converting from UTS PBR to UTS PBG

UTS Partition by Range
(Absolute page numbering
or
Relative page numbering**)**

ALTER TABLE … ALTER PARTITIONING TO PARTITION BY GROWTH

REORG table space

V13R1M507

Partition by Growth

```
ALTER TABLE E8071.TB01
      ALTER PARTITIONING TO    PARTITION BY GROWTH
  DSSIZE 8G MAXPARTITIONS 10
```

Note: The syntax is enforced at application compatibility level V13R1M507

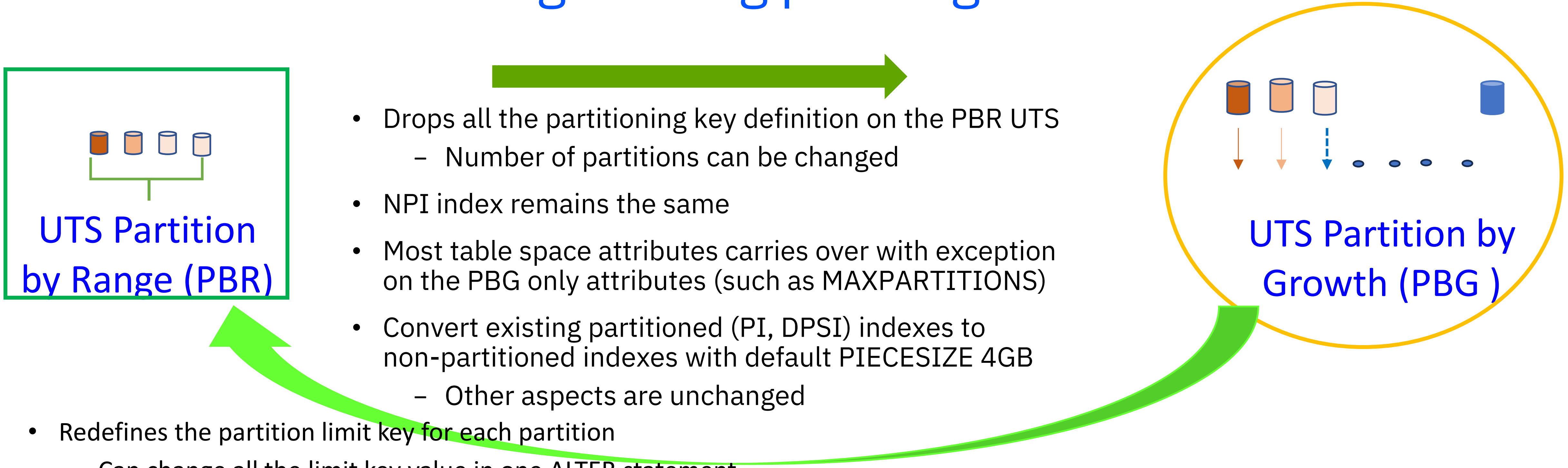
Stacking Support

- Stacking support of the following pending alter options with one REORG materializing all the changes
  - Table space level
    - MAXPARTITION
    - BUFFERPOOL
    - DSSIZE
    - SEGSIZE (exclude using SEGSIZE for conversion to UTS)
    - MEMBER CLUSTER
  - Table level
    - DROP COLUMN, ALTER COLUMN
      - ✓ Only apply to ALTER COLUMN is pending alter
      - ✓ Set ZPARM DDL_MATERIALIZATION = ALWAYS_PENDING
    - ALTER PARTITIONS
      - ✓ Only support syntax for **PBR->PBG, PBG->PBR or PBR->PBG->PBR** with one REORG materialization process
      - ✓ No support of the stacked **PBG->PBR->PBG**
  - Index level
    - BUFFERPOOL
    - COMPRESS attribute

Innovating partitioning changes

- Partition schema changes utilizing table space conversions within UTS
  - Db2 support PBR -- > PBG -- > PBR with one REORG materialization process
  - Achieve high availability partition schema changes
    - Partition limit key modification
      - ✓ Add column in the partition key
      - ✓ Single Alter statement requires rolling out the affected partition one at a time
      - ✓ Conversion enables ALTER limit key values of multiple partitions via a single ALTER statement
    - Drop partitions
      - ✓ Current DDL DROP PARTITION statement only supports the drop of the trailing partition
      - ✓ Conversion enables the dropping the middle of the partition with new partition scheme definitions
    - Removes the legacy 40-byte truncated limit key values
  - Benefits of new feature
    - Allows high availability with minimum impact to applications
    - Simplifies the process
    - Saves on valuable people resources

Partition schema changes using pending ALTER



- Drops all the partitioning key definition on the PBR UTS
  - Number of partitions can be changed
- NPI index remains the same
- Most table space attributes carries over with exception on the PBG only attributes (such as MAXPARTITIONS)
- Convert existing partitioned (PI, DPSI) indexes to non-partitioned indexes with default PIECESIZE 4GB
  - Other aspects are unchanged

- Redefines the partition limit key for each partition
  - Can change all the limit key value in one ALTER statement
- NPI index remains the same
- Most table space attributes carry over
- Note: If PI or DPSI is needed, it needs to be recreated
- One REORG materialized all the pending changes

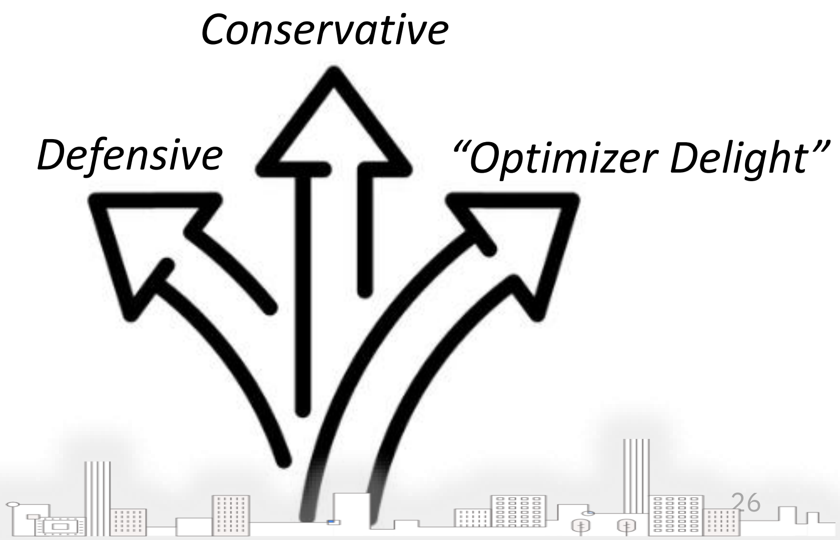
Function level : ***V13R1M507*** and application compatibility level: ***V13R1M507***

Summary of table space conversion within UTS

Table space type	Pending ALTER DDL	Target converted table space type
Partitioned by Range (absolute page numbering)	ALTER TABLESPACE …**PAGENUM RELATIVE**	Partitioned-by-Range relative page numbering (PBR RPN)
Partitioned by growth (PBG)	ALTER **TABLE** … **ALTER PARTITIONING TO PARTITION BY RANGE**	PBR RPN (V13R1M500)
Partitioned by Range (PBR RPN or PBR APN)	ALTER **TABLE** … **ALTER PARTITIONING TO PARTITION BY GROWTH**	Partitioned-by-growth (PBG)

Db2 REBIND

- When are Db2 REBINDS are necessary ?
  - After successfully migrating to a new Db2 release and running smoothly, progressively REBIND high used packages
    - Re-enable fast column processing
    - Avoid performance overhead of "puffing" code
    - Pickup latest runtime performance enhancements
    - Pickup latest maintenance to address issues previously seeded
  - Package invalidation following an online REORG to materialize an online schema change
    - NON-UTS to UTS conversions
    - UTS PBG to UTS PBR conversions
  - Exploit RELEASE(DEALLOCATE) optimizations
    - CICS-Db2 Protected threads
    - HP-DBATs
  - Elevate APPLCOMPAT level
  - After applying a Db2 preventative maintenance package
- *What is your appetite for Db2 access path change?*
  - Defensive
    - Adverse to change
  - Conservative
  - *"Optimizer Delight"*
    - Allow optimizer to choose at each REBIND



REBIND Options

- REBIND options
  - APREUSE (ERROR|WARN|NONE)
    - ERROR
      - ✓ Db2 tries to reuse the previous access paths for SQL statements in the package
      - ✓ Will guarantee the same access path or REBIND will fail
      - ✓ Db2 indicates the number of statements that cannot be reused in any package in a message
    - WARN
      - ✓ Db2 tries to reuse the previous access paths for SQL statements in the package
      - ✓ Successful with no warnings if same access path is available
      - ✓ If same access path is not available, optimizer will choose new access path (evaluated in previous step) and will be successful with warnings
    - NONE
      - ✓ Db2 does not try to reuse previous access paths for statements in the package

REBIND Options …

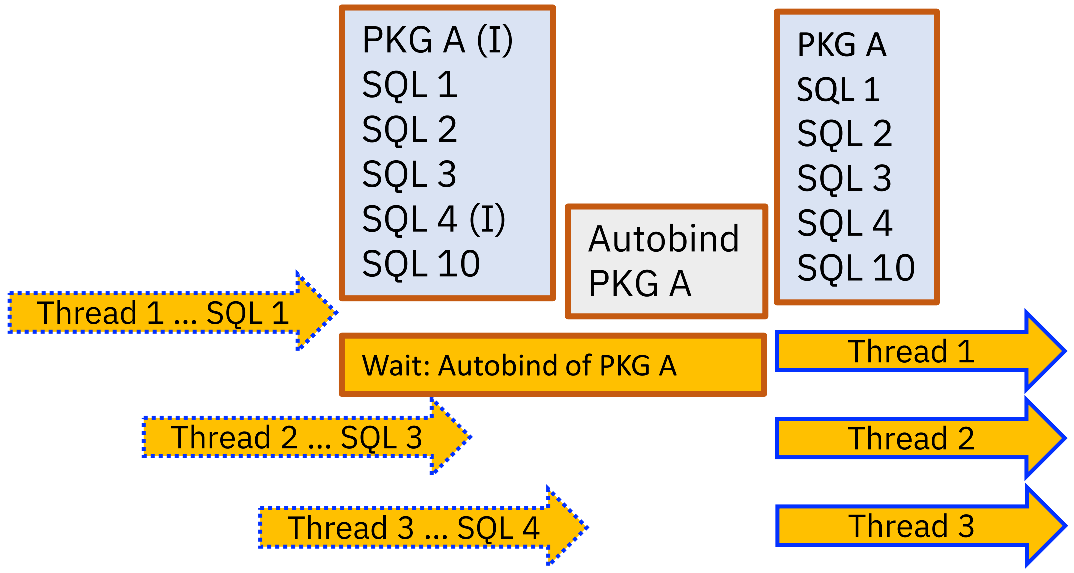
- REBIND options …
  - APCOMPARE (ERROR|WARN|NONE)
    - ERROR
      - ✓ Optimal access path will be selected (no guarantee the same access path will be selected)
      - ✓ If access path is structurally dissimilar when compare previous access path to current
        - REBIND will fail
    - WARN
      - ✓ Optimal access path will be selected (no guarantee the same access path will be selected)
      - ✓ If access path is structurally dissimilar compare previous access path to current
      - ✓ REBIND will be successful with warnings
        - Dissimilar SQL statements will be reported
    - NONE
      - ✓ Db2 does not try to reuse previous access paths for statements in the package
  - APREUSE = APCOMPARE = NONE
    - *“Optimizer Delight”*
      - Allow the optimizer to choose appropriate access path at each REBIND
      - Strongly recommend using Extended Plan Management
        - Revert a package to use previously saved access paths

REBIND Options …

- To minimize risk when performing a REBIND needed for non-access path operational improvements or when a Db2 package is invalidated
  - APREUSE(ERROR) should be the installation default
    - Defensive option to re-establish previous access path
    - Must apply APAR PH63063 (Db2 13)
      - APREUSE fails when attempting to reuse a Db2 12 access path that involves a view or table expression
    - Addressing failed APREUSE REBINDS
      - ✓ REBIND with APCOMARE(ERROR) EXPLAIN ONLY
        - Db2 13 APAR PH61970 supplies “Phase-In” support for REBIND package EXPLAIN ONLY
        - Will identify access path change
        - Evaluate access path differences
      - ✓ REBIND APREUSE(WARN)
        - Successful with no warnings if same access path is available
        - If same access path is not available, optimizer will choose new access path (evaluated in previous step) and will be successful with warnings
  - If successful REBIND using APREUSE (ERROR or WARN)
    - Develop a process to evaluate potential access path improvements
      - ✓ REBIND APCOMPARE(WARN) EXPLAIN ONLY
      - ✓ Perform analysis on any access path changes identified by the Db2 Optimizer
      - ✓ REBIND with APCOMPARE(WARN)
        - Optimal access path will be selected (no guarantee the same access path will be selected)

Package-Level Invalidation

- Package Invalidation
  - One or more SQL invalidations will result in the package being invalidated
    - Example
      - Package contains 10 SQL statements
      - One SQL invalidation will result in package (all 10 SQL statements) being marked invalid
  - Next request to execute will result in triggering a package autobind
    - Prior to V13R1M503
  - Package invalidation and subsequent autobind can significantly impact workload (application unavailability)
    - Execution must wait until autobind completes
    - Other executions must wait until initial autobind completes
    - Autobind failure (infrequent)
      - Package is marked inoperative
      - Explicit rebind is necessary



Statement-Level Invalidation

- Reducing the impact of an AUTOBIND operation …
  - Previous behavior
    - Db2 tracks application dependencies at package level
    - An operation on any object requiring invalidation results in the entire package marked as invalid; even when only a subset of SQL statements in that package needs to be invalidated
    - This is broad and limits Db2 flexibility to enhance and improve invalidation processing
  - Db2 13 behavior
    - Provide more granular dependency & validity tracking infrastructure, laying foundation for enhancements such as reduced impact of invalidated packages and improved DDL & static DML concurrency
    - New DEPLEVEL BIND/REBIND option determines recording of statement level dependencies in addition to package level dependencies.
    - New system parameter PACKAGE_DEPENDENCY_LEVEL (SPRMPKGDEPLVL) sets DEPLEVEL default
    - New catalog tables SYSPACKSTMTCOPY and SYSPACKSTMTDEP

Statement-Level Invalidation ...

FL 500

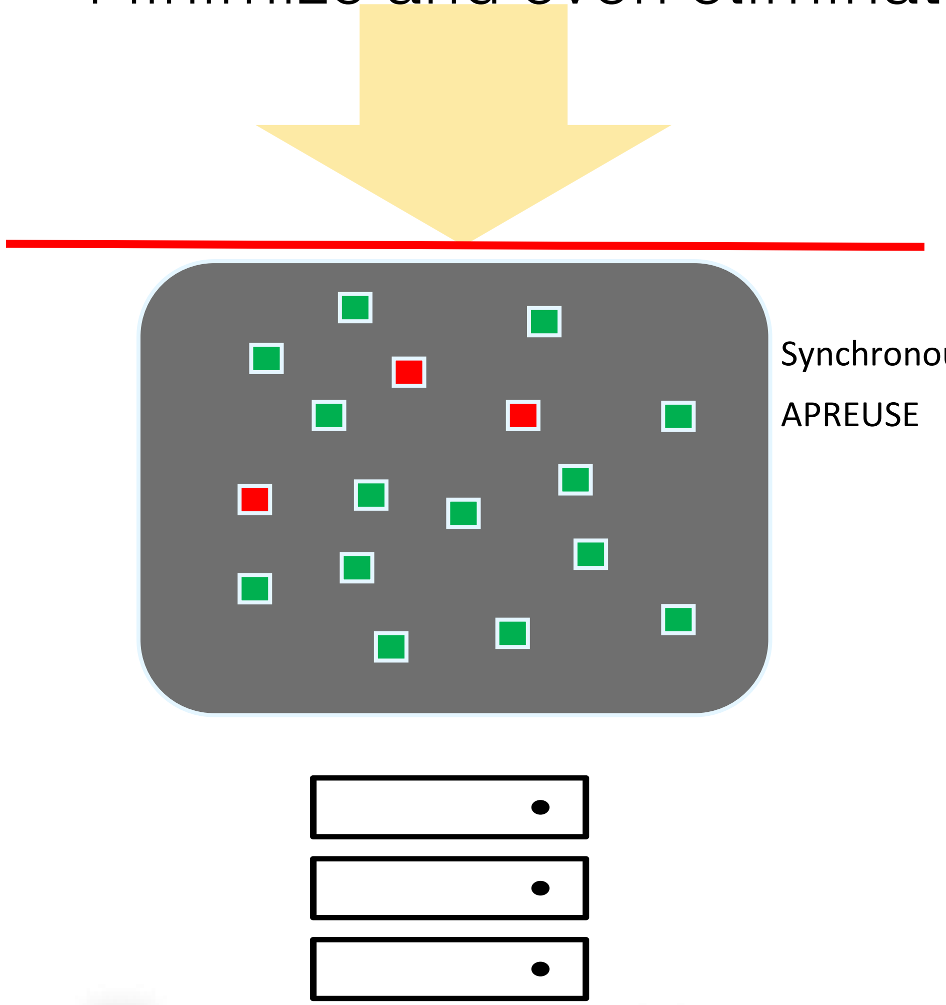
FL 502

FL 504

- Reducing the impact of an AUTOBIND operation
  - FL500: CATMAINT can be executed to take catalog level to V13R1M501 level
    - One of the new tables in V13R1M501 catalog is SYSPACKSTMTDEP
  - FL502: packages can bound/rebound with new DEPLEVEL(STATEMENT) option
    - That causes statement-level dependencies to be recorded in SYSPACKSTMTDEP
  - FL504: if ALTER causes invalidation of package bound with DEPLEVEL(STATEMENT), next request to execute package still triggers autobind, BUT…
    - Autobind done in background, and package can still be executed even before autobind completes: non-invalidated statements execute as usual, invalidated statements incrementally bound when executed
    - When autobind completes, newly-regenerated package phased in (similar to rebind phase-in functionality of Db2 12 FL505)
    - If autobind fails, package gets “advisory rebind” status and can still be executed (non-invalidated statements execute as usual, invalidated statements incrementally bound when executed) – explicit rebind will put package back in valid state

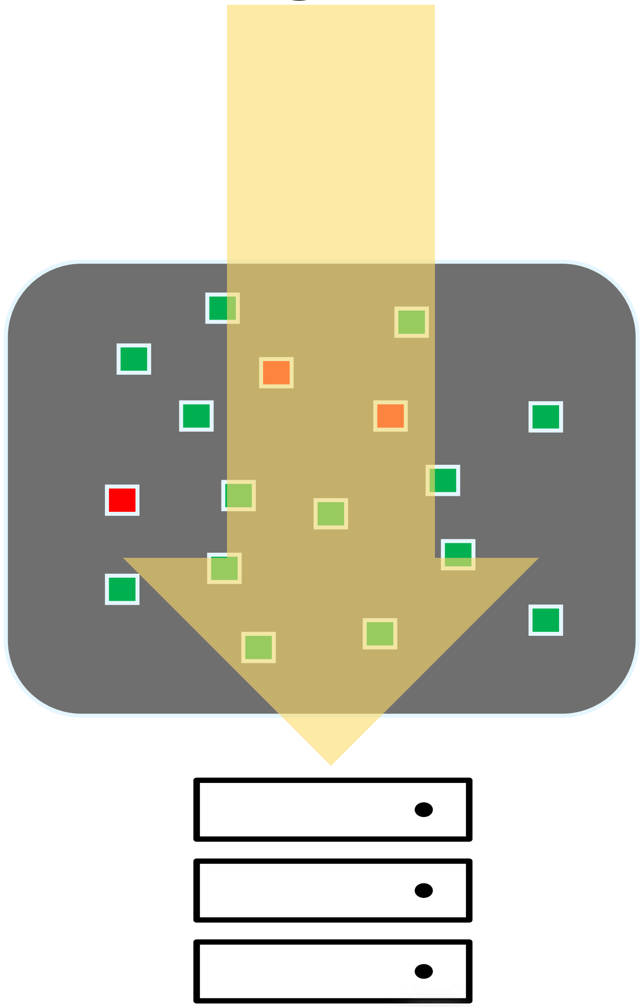
Statement-Level Invalidation ...

- Minimize and even eliminate impact from package invalidation



Synchronous autobind

APREUSE



Incremental bind

Asynchronous autobind

Phase-in APREUSE

IDUG

2026

Sydney | March 16 - 18

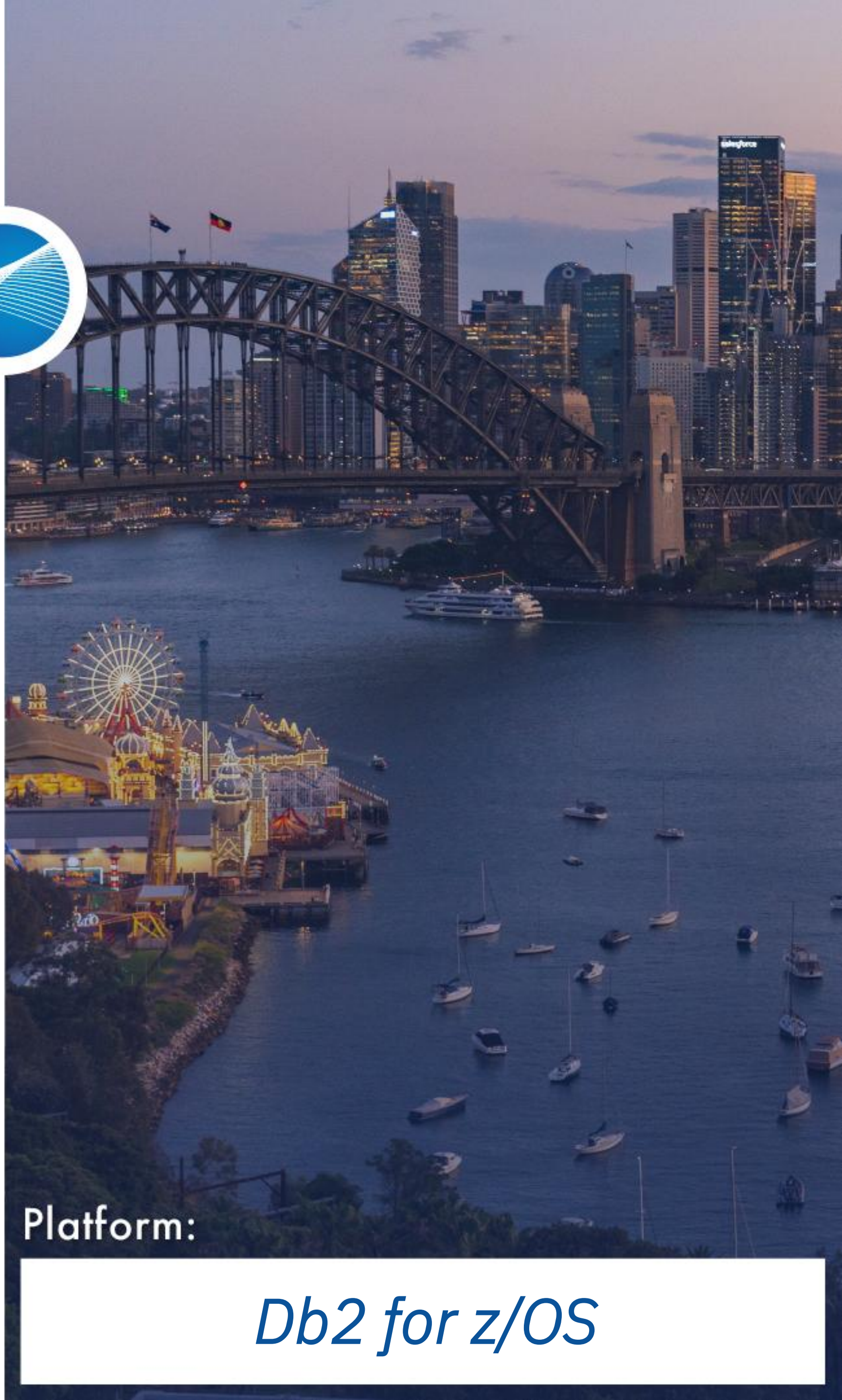
AU Db2 TECH CONFERENCE

Improved Availability through the Evolution of Universal Table Space (UTS)

Anthony Ciabattoni, IBM

Contact: aciabattoni@ibm.com

Session Code: A08



Platform:

Db2 for z/OS

Thank You

Questions

IDUG

2026 Australia **Db2** Tech Conference