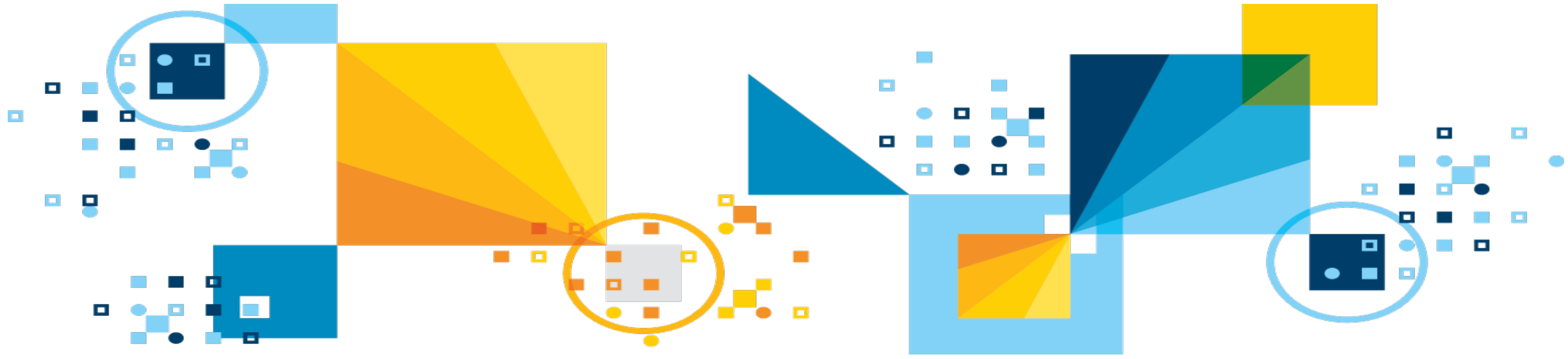
Robert Catterall
IBM Senior Consulting Db2 for z/OS Specialist
rfcatter@us.ibm.com

# The most important stuff you should know about Db2 12 for z/OS

Tridex Db2 Users Group
September 10, 2020



## Agenda

- Delivery of new functionality
- Exploiting the Z “stack”
- Data sharing
- Physical database design
- Performance
- SQL
- Non-disruptive rebind
- Utilities

## Delivery of new functionality

## Continuous delivery

- Instead of delivering a huge batch of new features every 3 years through a new Db2 version, we are delivering smaller batches of new functionality 3-4 times per year via new Db2 12 function levels
  - Information about function levels:
    https://www.ibm.com/support/knowledgecenter/en/SSEPEK_12.0.0/wnew/src/tpc/db2z_db2functionlevels.html
- From perspective of a Db2-using organization, continuous delivery begins once you’ve gotten to Db2 12 function level V12R1M500 (analogous to what was formerly known as new-function mode)
  - When you first come up at Version 12 after migrating from Db2 11, the function level is V12R1M100 – analogous to what was formerly known as conversion mode

## More on continuous delivery

- The code that enables new function levels is provided via the Db2 maintenance stream, in the form of fixes
  - That does NOT mean that you should alter your regular schedule for upgrading Db2 maintenance – keep doing that as before (twice a year, quarterly, whatever)
  - Why you don't need to be concerned about Db2 code going to new level when you do regular quarterly (for example) upgrade of Db2 maintenance: new code level will not affect your system until you activate associated function level
    - In other words, updating Db2 12 code to support a new function level (via application of a fix) and activating the new features of a function level (via execution of the new command ACTIVATE FUNCTION LEVEL) *are two separate actions*

# CATMAINT may be required for new function level…

| | TO V12R1- | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| FROM | M100 | M500 | M501 | M502 | M503 | M504 | M505 | M506 | M507 |
| V11 | CATM | N | N | N | N | N | N | N | N |
| V12R1M100 | | Y | Y | N | N | N | N | N | N |
| V12R1M500 | Y* | | Y | CATM | CATM | Y^CATM | CATM | Y^CATM | CATM |
| V12R1M501 | Y* | Y* | | CATM | CATM | Y^CATM | CATM | Y^CATM | CATM |
| V12R1M502 | Y* | Y* | Y* | | CATM | Y^CATM | CATM | Y^CATM | CATM |
| V12R1M503 | Y* | Y* | Y* | Y* | | Y | CATM | Y^CATM | CATM |
| V12R1M504 | Y* | Y* | Y* | Y* | Y* | | CATM | Y^CATM | CATM |
| V12R1M505 | Y* | Y* | Y* | Y* | Y* | Y* | | Y | CATM |
| V12R1M506 | Y* | Y* | Y* | Y* | Y* | Y* | Y* | | CATM |
| V12R1M507 | Y* | Y* | Y* | Y* | Y* | Y* | Y* | Y* | |

When CATMAINT has to be run to enable activation of a new function level, it is executed with the new option CATMAINT UPDATE LEVEL

Example: CATMAINT UPDATE LEVEL(V12R1M502)

CATM = CATMAINT required for FL
Y^CATM = CATMAINT for previous FL required
* = went back from higher FL

# -DISPLAY GROUP information

(relevant to standalone as well as data sharing Db2 subsystems)

Catalog level

```
-DIS GROUP
DSN7100I  -DB2A DSN7GCMD
*** BEGIN DISPLAY OF GROUP(DSNCAT   ) CATALOG LEVEL(V12R1M500)
                  CURRENT FUNCTION LEVEL(V12R1M500)
                  HIGHEST ACTIVATED FUNCTION LEVEL(V12R1M500)
                  HIGHEST POSSIBLE FUNCTION LEVEL(V12R1M500)
                  PROTOCOL LEVEL(2)
                  GROUP ATTACH NAME(DSNG)
-------------------------------------------------------------
DB2          SUB                     DB2    SYSTEM    IRLM
MEMBER   ID  SYS  CMDPREF   STATUS   LVL    NAME      SUBSYS
-------- --- ---- --------  -------- ------ --------  ----
DB2A       1 DB2A -DB2A     ACTIVE   121500 SVTEC02   PR21
DB2B       2 DB2B -DB2B     ACTIVE   121500 SVTEC02   RRLM
-------------------------------------------------------------
...
```

Function levels

Code level

## APPLCOMPAT in a Db2 12 environment

- Still has a twofold purpose:
  1. Enables program to utilize functionality of a given Db2 12 function level
  2. Protects program from "SQL incompatibility" (same SQL, same data, different result) that could be introduced by Db2 version or function level
     - **Example:** starting with Db2 11 in new-function mode, an 8-byte store clock value is no longer a valid argument in a CAST(… AS TIMESTAMP) expression – if that change would "break" a program, you can bind its package with APPLCOMPAT(V10R1)
- In a Db2 12 system, acceptable APPLCOMPAT values are V10R1, V11R1 and any Db2 12 function level (e.g., V12R1M501)
- APPLCOMPAT in ZPARM: default value used when APPLCOMPAT not specified for BIND PACKAGE (or is "not there" for REBIND PACKAGE)
- Db2 12: APPLCOMPAT applies to DDL as well as DML statements

## A little more on Db2 12 and APPLCOMPAT

- Recommendation: soon after coming up with Db2 12 at the V12R1M100 function level, rebind all plans and all packages
  - Db2 12 optimizer enhancements available, and Db2 12-generated package code likely more CPU-efficient than Db2 11-generated code (even when access paths don't change)
  - When you do that large-scale rebind of packages in Db2 12 system, you <u>can</u> change APPLCOMPAT specification (e.g., to V12R1M100), but you don't have to
    - For one thing, access path selection is NOT affected by a package's APPLCOMPAT value
- What about after that?
  - Especially for <u>static</u> SQL packages, generally no need to change APPLCOMPAT value as you activate new Db2 12 function levels – again, <u>can</u> do that, but don't have to
  - Probably want to change APPLCOMPAT in ZPARM after you activate new function level
    - Changing ZPARM will not cause APPLCOMPAT to change for a package when rebound, *assuming package <u>has</u> an APPLCOMPAT value*

## What about IBM Data Server Driver / Db2 Connect packages?

- Personally, I would want to rebind packages in NULLID collection with new APPLCOMPAT value soon after activating a new Db2 12 function level
  - Reason: I would like for developers of programs that use these packages (e.g., programs that use JDBC, ODBC) to have access to latest SQL functionality
- What if you do that, but a DRDA requester application needs SQL behavior of an earlier Db2 level because of some SQL incompatibility issue?
  - You can use the Db2 profile tables to direct such an application to an alternate IBM Data Server Driver / Db2 Connect collection in which packages are bound with a lower APPLCOMPAT value (see next slide)
  - Same approach could be used if you bind NULLID packages with a lower APPLCOMPAT value and you want to direct an application to an alternate collection in which those packages are bound with a higher APPLCOMPAT value

## Pointing DRDA requester to "not NULLID" collection

- How to do that using Db2 profile tables:
  - Insert a row for the application in SYSIBM.DSN_PROFILE_TABLE
  - In associated row in SYSIBM.DSN_PROFILE_ATTRIBUTES, indicate that Db2 is to automatically issue SET CURRENT PACKAGE PATH = *coll-id* for the application, where *coll-id* is the name of the alternate collection
  - Not just for APPLCOMPAT – packages in alternate collection could be bound with some other option that is different versus NULLID (e.g., RELEASE(DEALLOCATE))
    - You could have 3, 4 or more different collections of IBM Data Server Driver / Db2 Connect packages, bound with different options to get different behaviors – no problem
  - More information on this approach:

    http://robertsdb2blog.blogspot.com/2019/07/db2-for-zos-talking-about-applcompat.html

    http://robertsdb2blog.blogspot.com/2018/07/db2-for-zos-using-profile-tables-to.html

## Exploiting the Z “stack”

## Function level 502: ZPARM, DDL for z/OS data set encryption

- Prior to Db2 12 function level 502: Db2 people had to request of RACF or storage admins that certain Db2 data sets be encrypted
- Function level 502 and above: Db2 people can say to RACF or storage admins, “give us some key labels that we can use for Db2 data sets, and we’ll take it from there”

**Db2 11, or Db2 12 pre-FL502**

↓

Native z/OS options:

1. RACF data set profile
2. Explicit definition
   - Active logs: IDCAMS
   - Utility data sets: JCL DSKEYLBL
   - User managed objects: IDCAMS
3. SMS data class

**Db2 12 FL502 or higher**

↓

New Db2 specifications

- ZPARM **ENCRYPTION_KEYLABEL**
  - Encrypt new catalog/directory objects
  - Encrypt new archive log data sets
- SQL ALTER/CREATE with TABLE or STOGROUP
  - New option **KEY LABEL**
- Utilities, including Online REORG, allocate the new data sets as encrypted
- Display options, like:
  - -DIS GROUP or REPORT utility or -DIS LOG or -DIS ARCHIVE

## More on Db2 and z/OS data set encryption

- Functionality provided by DFSMS
  - Data encrypted on write, decrypted on read (so, application transparent)
- Encryption performance good on z13, great on z14 (up to 7X more CPU-efficient on z14 versus z13), really great on z15
  - Even more efficient with large Db2 buffer pools: bigger pool = fewer disk reads, leading to reduced decrypt cost because data decrypted when read into memory
  - Encryption of Db2 data “at scale” now very feasible
- When existing object encrypted, encryption done via online REORG
- Archive log data sets encrypted when created (ZPARM: ENCRYPTION_KEYLABEL)
- If encryption of active log data sets desired: create new encrypted log data sets, REPRO data from old to new log data sets, change log inventory

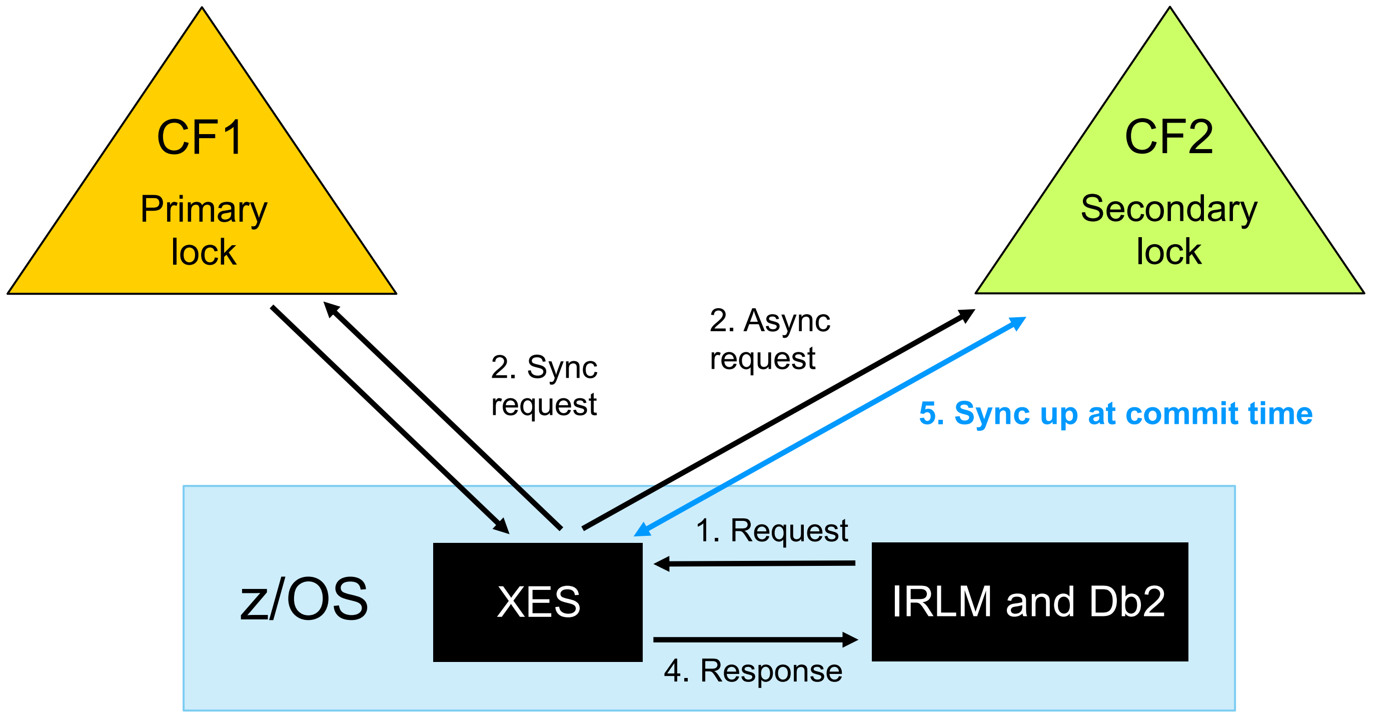
## Data sharing

# Asynchronous lock structure duplexing

- Challenge: if you have 2-box Parallel Sysplex, and CEC housing CF LPAR holding lock structure fails, you have simultaneously lost lock structure and at least one member of Db2 data sharing group
  - Lock structure rebuild will fail, because that process requires information from all members of data sharing group, and at least 1 member lost when CEC failed
  - Can't do data sharing without a lock structure, so whole data sharing group fails
- One way to eliminate possibility of this "double failure" scenario: duplex the lock structure
  - Problem with that (historically): very high overhead, due to "brute force" mode of duplexing: <u>every</u> request to primary lock structure also driven <u>synchronously</u> to secondary

## Db2 12: asynchronous lock structure duplexing

1. IRLM requests lock
2. Lock request sent synchronously to primary CF, asynchronously to secondary CF
3. Response returned to XES
4. XES responds to IRLM
5. Sync up at commit time



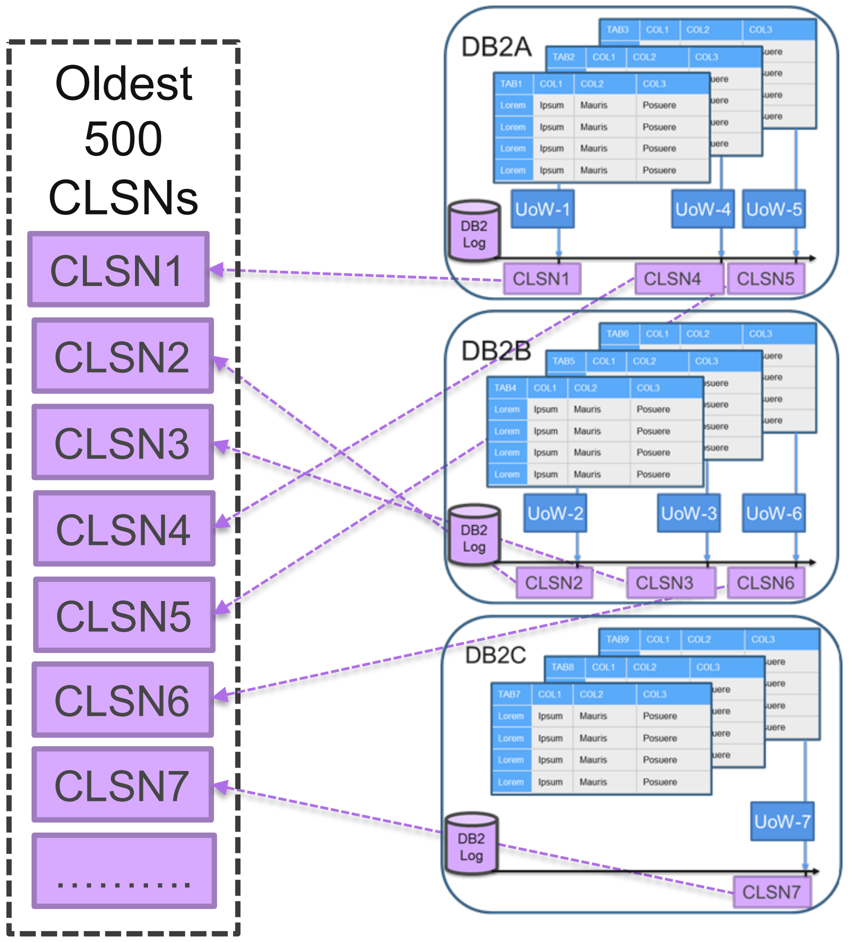
CPU overhead only a little greater vs. lock structure in simplex mode

## Elimination (virtually) of a lock avoidance penalty

- Db2 has 2 indicators it can use to determine if data in page is in committed state – if that can be verified, data can be read without having to S-lock page or row
  - Those two indicators are PUNC bit and CLSN (commit log sequence number)
  - Every page set and partition has its own CLSN – the log point at which longest-running still-in-flight data-changing unit of work accessing the object got started
  - In every page there is log point value at which data in page last changed – if CLSN for object is greater than that page-level log point value, data in page can be read without locking page (or row)
- In the past, in data sharing system a single CLSN (group CLSN) used for all GBP-dependent data sets – that CLSN is “oldest” of all CLSNs
  - Result: ONE really long-running data-changing unit of work negatively impacts lock avoidance for just about all processes in the system

# Db2 12: lock avoidance penalty virtually eliminated

- Db2 caches, in memory of members and in shared communications area, the 500 oldest CLSNs in the data sharing system
- At worst, for an object not associated with any of the 500 oldest CLSNs, the newest of the 500 oldest CLSNs will be used for that object for lock avoidance purposes
- Result: more lock avoidance versus Db2 11, meaning fewer global lock requests propagated to lock structure in coupling facility, meaning enhanced CPU efficiency



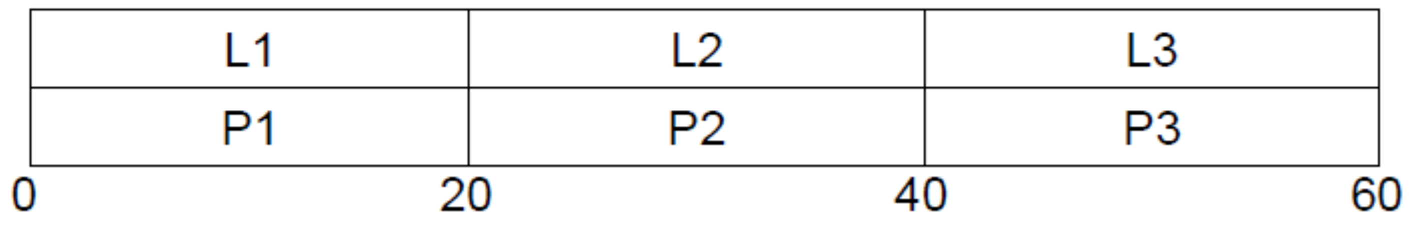
## Physical database design

## Insert partition into middle of PBR table space

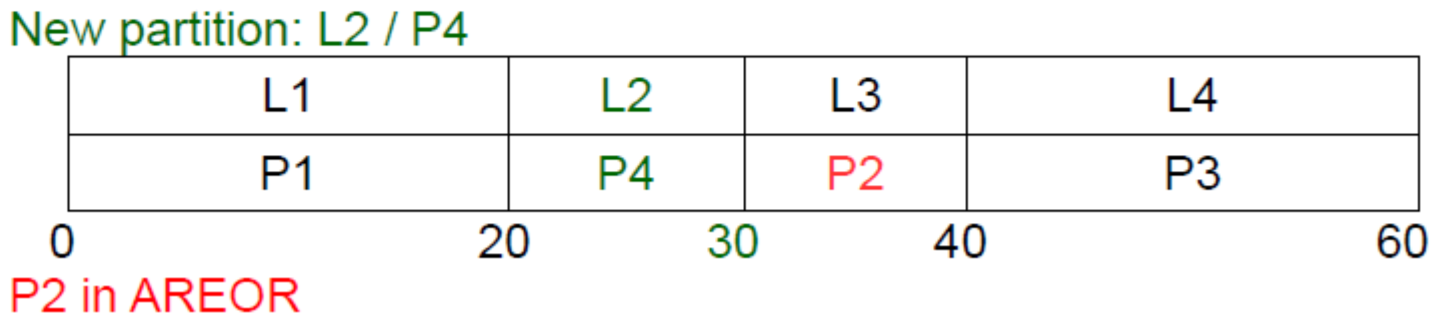
- ALTER table ADD PARTITION **ENDING AT (30)** **ALTER PARTITION** (2) **ENDING AT** (40)

  This part of statement is optional

  – Before ALTER:



  – After ALTER:



## More on insert-partition-into-middle

- Table space in question has to be <u>universal</u> PBR
- Even though only partition that gets smaller (as result of new partition being added) is in AREOR state after ALTER, and only that partition is REORGed to materialize change, <u>all</u> table space partitions briefly drained at end of REORG
  - This because of the re-mapping of logical to physical partitions necessitated by the addition of partition in the middle of the table space

## A new type of universal PBR table space

- Some limitations of traditional universal PBR table spaces:
  - Max of 16 TB of data (if using 4K pages)
  - Max number of partitions goes down as DSSIZE goes up, page size goes down
    - Example: if DSSIZE = 256 GB and page size = 4K, maximum number of partitions is 64
- Limitations exist because, for traditional universal PBR table space, every page is uniquely numbered in continuously ascending fashion
  - This is referred to as absolute page numbering
- Db2 12 introduces new type of universal PBR table space that uses relative page numbering (so new type called an RPN table space)
  - The key: we start over with page numbering for each succeeding partition
  - Implication: page number alone does not uniquely identify a page – combination of page number and partition number uniquely identifies page

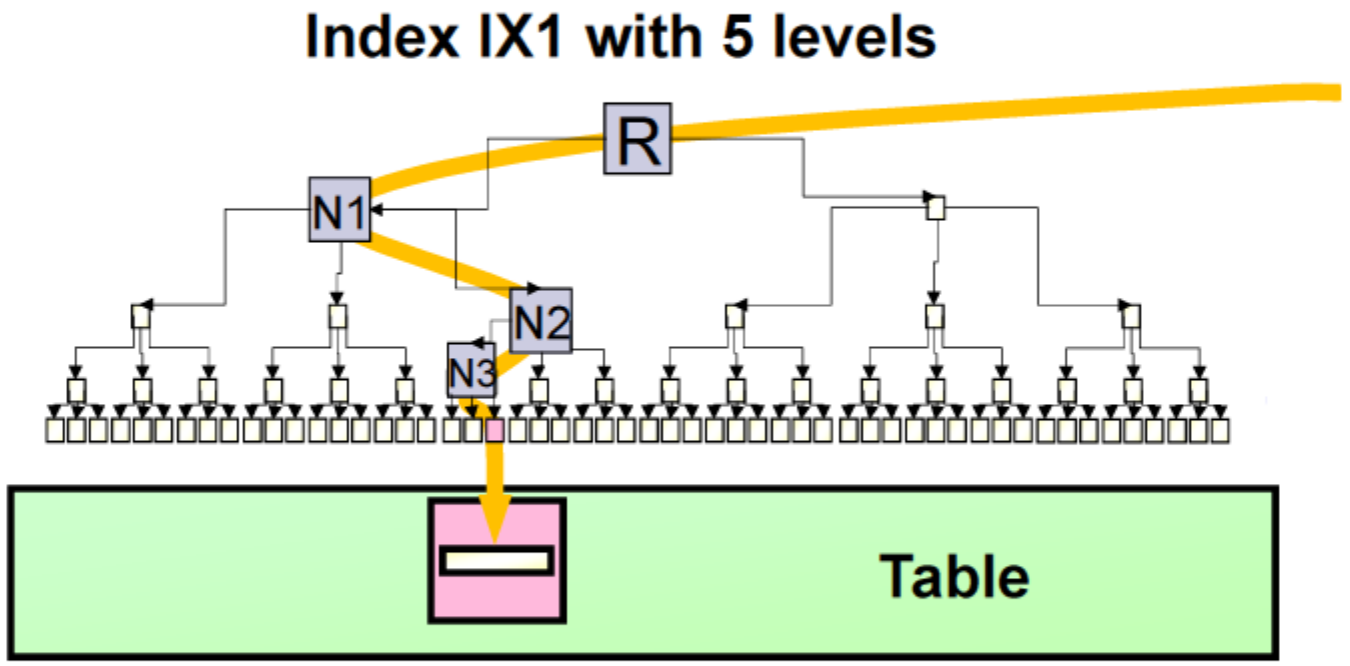
## RPN opens up all kinds of new capabilities

- Up to 4096 partitions, even with large DSSIZE and small page size
- DSSIZE can be nG, with “n” being any integer between 1 and 1024
  - Want DSSIZE for a partition to be 173G? No problem
- With 4096 partitions of 1024G each: up to 4096 TB of data in 1 table
  - And, max rows in a table goes to 280 trillion
- And there’s more:
  - With RPN, different partitions of a table space can have different DSSIZE values
  - And increase in DSSIZE for partition is immediate change – no REORG needed
- Note: an existing universal PBR table space can be changed to RPN via ALTER with PAGENUM RELATIVE, followed by online REORG of table space

## Performance

## Addressing an index-related performance penalty

- As table grows, indexes defined on table go to additional levels
- Each level added to an index adds a GETPAGE to every index “probe”
- More GETPAGEs lead to increased CPU cost of query execution

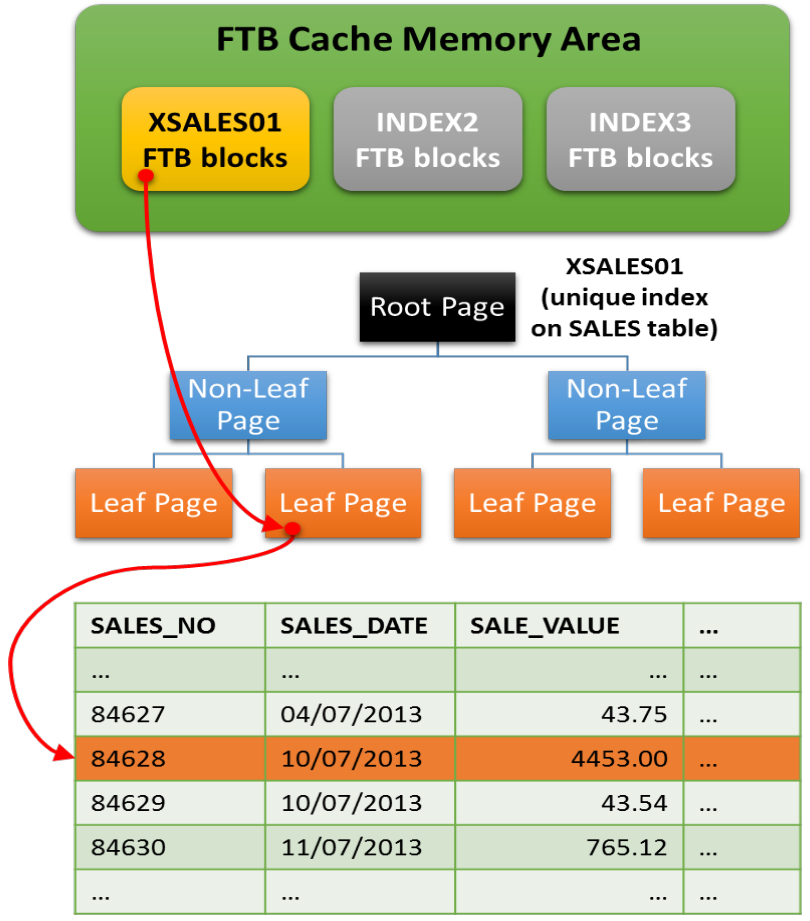


*Accessing table row via 5-level index requires 6 GETPAGEs - 5 of which are index-related*

# Db2 12 index fast traverse blocks (FTBs)

- Db2 12 can build FTB structures for certain indexes (index has to be unique, key length cannot exceed 64 bytes)
  - SYSINDEXCONTROL catalog table lets you make decisions regarding indexes and FTBs, but recommended to let DB2 handle this
- Example of CPU efficiency improvement:
  - Suppose query has an "equals" predicate that matches an index for which an FTB has been built by Db2
  - Db2 can go to the FTB with the value specified in the query predicate, and FTB identifies the index leaf page in which the value is located
  - If index has 5 levels, so that 6 GETPAGEs formerly needed to access data row (5 index GETPAGEs, 1 table space GETPAGE), now only 2 GETPAGEs (1 for index leaf page, 1 for table page holding row)
- By default, size of area in Db2 DBM1 address space available for FTBs is 20% of the size of subsystem's buffer pool configuration
  - If buffer pool configuration size = 100 GB, size of FTB area is 20 GB

```
SELECT SALES_DATE, SALE_VALUE ...
FROM SALES
WHERE SALES_NO = 84628
```



| SALES_NO | SALES_DATE | SALE_VALUE | ... |
|---|---|---|---|
| ... | ... | ... | ... |
| 84627 | 04/07/2013 | 43.75 | ... |
| 84628 | 10/07/2013 | 4453.00 | ... |
| 84629 | 10/07/2013 | 43.54 | ... |
| 84630 | 11/07/2013 | 765.12 | ... |
| ... | ... | ... | ... |

## SQL

# Old MERGE statement: improvement needed

Pre-Db2 12
MERGE statement

```
MERGE INTO ACCOUNT AS A
USING (VALUES (:hv_id, :hv_amount)
FOR 3 ROWS)
AS T (ID, AMOUNT)
ON (A.ID = T.ID)
WHEN MATCHED
 THEN UPDATE SET BALANCE = A.BALANCE + T.AMOUNT
WHEN NOT MATCHED THEN INSERT (ID, BALANCE)
VALUES (T.ID, T.AMOUNT)
NOT ATOMIC CONTINUE ON SQLEXCEPTION;
```

Input can only be host variable arrays or a list of values

Only very simple “matched” and “not matched” clauses can be specified

- Only one update and one insert action can be specified
- Delete is not an option

Required when there are multiple “rows” of input values – the rows are processed separately, and processing continues if errors are encountered

AND, a target table row can be operated on multiple times in the execution of one MERGE statement – not always a desired behavior

# Db2 12 MERGE: ease of use, flexibility

```
MERGE INTO RECORDS AR
USING (SELECT ACTIVITY, DESCRIPTION, DATE, LAST_MODIFIED
FROM ACTIVITIES_GROUPA) AC
ON (AR.ACTIVITY = AC.ACTIVITY) AND AR.GROUP = 'A'
WHEN MATCHED AND AC.DATE IS NULL THEN SIGNAL SQLSTATE '70001'
SET MESSAGE_TEXT =
AC.ACTIVITY CONCAT ' CANNOT BE MODIFIED. REASON: DATE IS NOT KNOWN'
WHEN MATCHED AND AC.DATE < CURRENT DATE THEN DELETE
WHEN MATCHED AND AR.LAST_MODIFIED < AC.LAST_MODIFIED THEN
UPDATE SET
(DESCRIPTION, DATE, LAST_MODIFIED) = (AC.DESCRIPTION, AC.DATE,
DEFAULT)
WHEN NOT MATCHED AND AC.DATE IS NULL THEN SIGNAL SQLSTATE '70002'
SET MESSAGE_TEXT =
AC.ACTIVITY CONCAT ' CANNOT BE INSERTED. REASON: DATE IS NOT KNOWN'
WHEN NOT MATCHED AND AC.DATE >= CURRENT DATE THEN
INSERT (GROUP, ACTIVITY, DESCRIPTION, DATE)
VALUES ('A', AC.ACTIVITY, AC.DESCRIPTION, AC.DATE)
ELSE IGNORE;
```

New input options: table, view, fullselect

Can use SIGNAL to provide customized error codes and messages

DELETE is now an option

Can have multiple "matched" and "not matched" clauses with additional predicates, enabling multiple update, delete, and insert actions

New IGNORE option: when input row does not meet any "matched" or "not matched" conditions, ignore it

## Piece-wise DELETE

- Pre-Db2 12 problem: DELETE such as the one below could affect a very large number of rows in a table

```
DELETE FROM T1 WHERE C1 > 7
```

  – Many locks could be acquired (or lock escalation could occur), and huge amount of data could written to log (backout would take a long time)

- Db2 12 enables “piece-wise” DELETE (break big DELETE into smaller parts) via support for FETCH FIRST clause in DELETE statement

```
DELETE FROM T1 WHERE C1 > 7 FETCH FIRST 5000 ROWS ONLY;
COMMIT;
DELETE FROM T1 WHERE C1 > 7 FETCH FIRST 5000 ROWS ONLY;
COMMIT;
```

Delete first chunk of rows

Delete 2nd chunk of rows

  – Available with function level 500

## A new kind of trigger

- What were called "triggers" before Db2 12 are called "basic triggers" in a Db2 12 environment
- Db2 12 "advanced triggers" deliver multiple benefits, including:
  - Body of advanced trigger can include SQL PL routine with logic flow control statements such as IF, ITERATE, LOOP, and WHILE
    - Enables easier creation of more highly-functional triggers (previously, advanced functionality might require trigger to call stored procedure)
  - Provides compatibility with Db2 for Linux/UNIX/Windows
  - Multiple versions of a given advanced trigger can be defined and maintained
    - This solves a problem that formerly could arise when several triggers have been defined for a table (see next slide)
  - Available with function level 500

# Change advanced trigger and preserve “firing order”

- Problem (before advanced triggers):
  - If multiple triggers defined on table have same activation time (e.g., AFTER) and are “fired” (i.e., activated) by same event, order of firing depends on order of creation
  - Changing a trigger requires drop/re-create, and that could change firing order unless ALL relevant triggers are dropped/re-created in desired order
- Advanced triggers – problem solved!
  - Now, <u>3 options</u> for changing existing trigger <u>without affecting firing order</u>:

New syntax

ID of version being changed

```
CREATE OR REPLACE TRIGGER trigger-name VERSION version-ID…
```

New syntax

ID of version being changed

```
ALTER TRIGGER trigger-name REPLACE VERSION version-ID…
```

New syntax

```
ALTER TRIGGER trigger-name ADD VERSION version-ID…
ALTER TRIGGER trigger-name ACTIVATE VERSION version-ID…
```

<u>New</u> version ID

## Example of an advanced trigger

As with other SQL PL routines, debug with Data Studio

These are among new CREATE TRIGGER options

SQL PL logic flow control statements

**Trigger body contains this logic:**

If class end time is null, value is set to 1 hour after start of class; otherwise, if class ends after 9pm then an error is returned

```
CREATE TRIGGER MYTRIG01
BEFORE INSERT ON MYTAB
REFERENCING NEW AS N
FOR EACH ROW
ALLOW DEBUG MODE
QUALIFIER ADMF001
WHEN(N.ending IS NULL OR n.ending > '21:00')
L1: BEGIN ATOMIC
        IF (N.ending IS NULL) THEN
            SET N.ending = N.starting + 1 HOUR;
        END IF;
        IF (N.ending > '21:00') THEN
            SIGNAL SQLSTATE '80000'
            SET MESSAGE_TEXT =
                    'Class ending time is beyond 9 pm';
        END IF;
        SET GLOBAL_VAR = NEW.C1;
END L1#
```

With advanced trigger, SET is not restricted to transition variables – it can also be used with global variables and SQL variables (latter refers to variables declared in body of trigger)

## Non-disruptive rebind

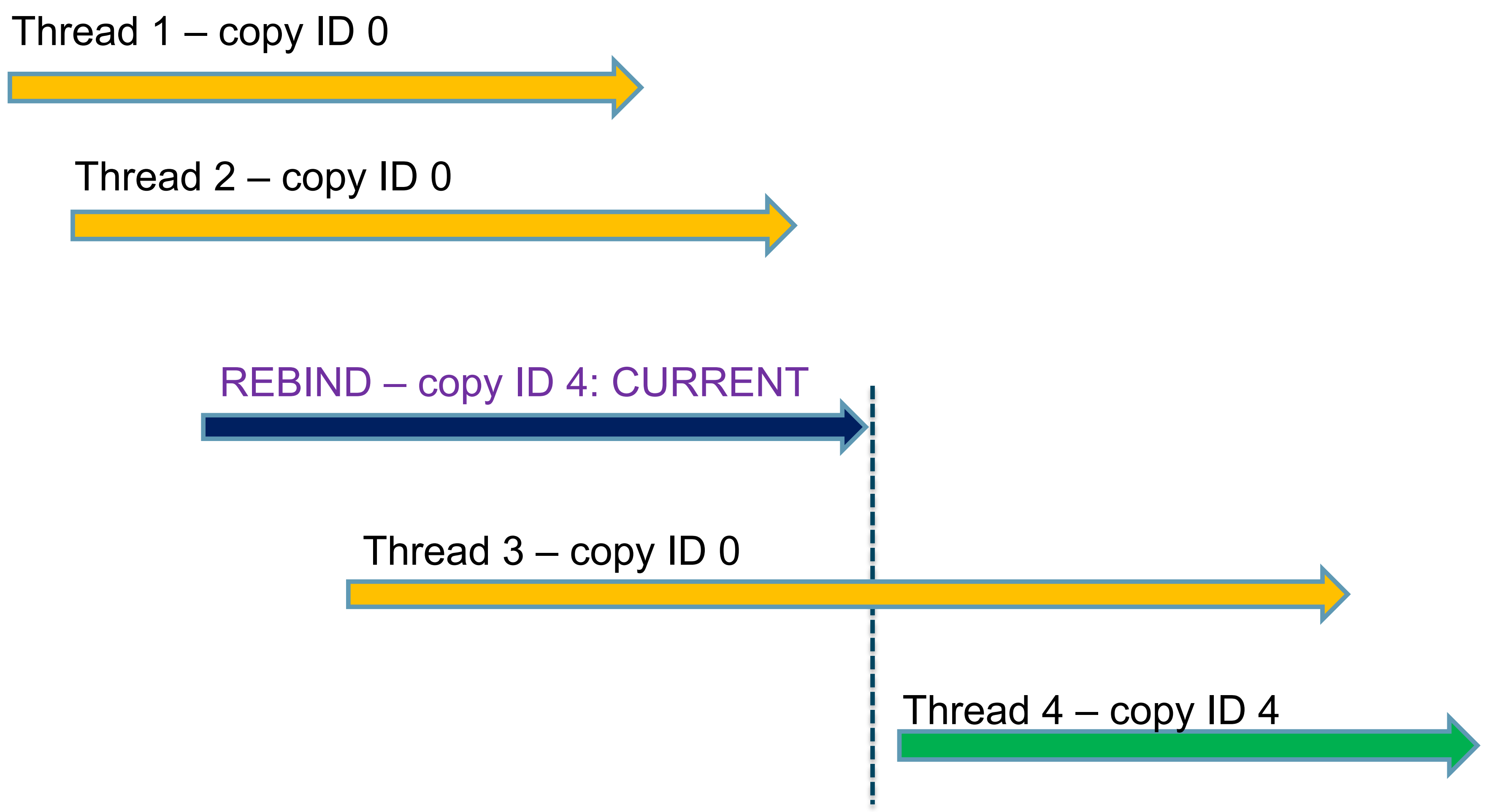
## Sometimes package rebind is problematic

- Can’t rebind package if “in-use”
  - RELEASE(DEALLOCATE) package allocated to persistent thread (thread that persists through commits) considered in-use until thread is terminated
  - Even when package bound with RELEASE(COMMIT), may be executed with such frequency that use count does not go to zero
  - Even with bind/rebind “break in” functionality introduced with Db2 11, attempt to rebind package may time out, and/or rebind action may disrupt workload
- Solution delivered with Db2 12 function level M505: rebind phase-in

## How rebind phase-in works

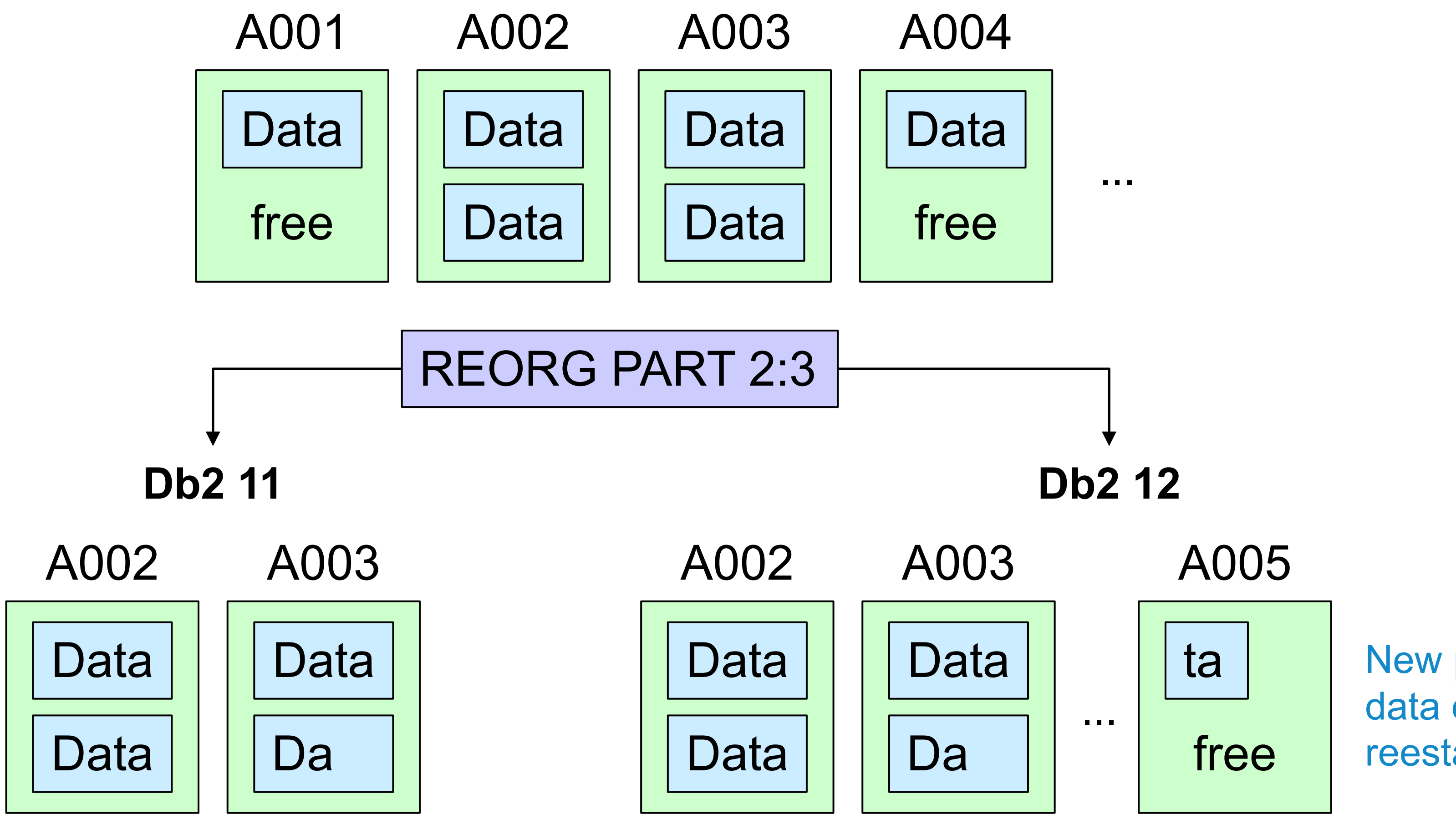
- Builds on Db2 for z/OS plan management functionality, by which up to three “copies” of each package are retained: current, previous and original (if you have default PLANMGMT = EXTENDED in ZPARM)
  - Historically, current package has been designated as copy 0, previous as copy 1, original as copy 2
- Re-bind phase-in: *when package to be rebound is in-use,* copy number of new package will be next value in series 0, 4, 5, 6…, 16, relative to copy number of formerly current package, with wrap-around at 16
  - Gap between 0 and 4 because 1 and 2 are previous and original, 3 is reserved
  - Threads using formerly current package during rebind are not impacted
  - Threads to which package allocated following rebind will get new package copy
  - Formerly current package will become new previous copy (number 1)

# REBIND phase-in: get the picture?



## Utilities

## REORG subset of PBG table's partitions



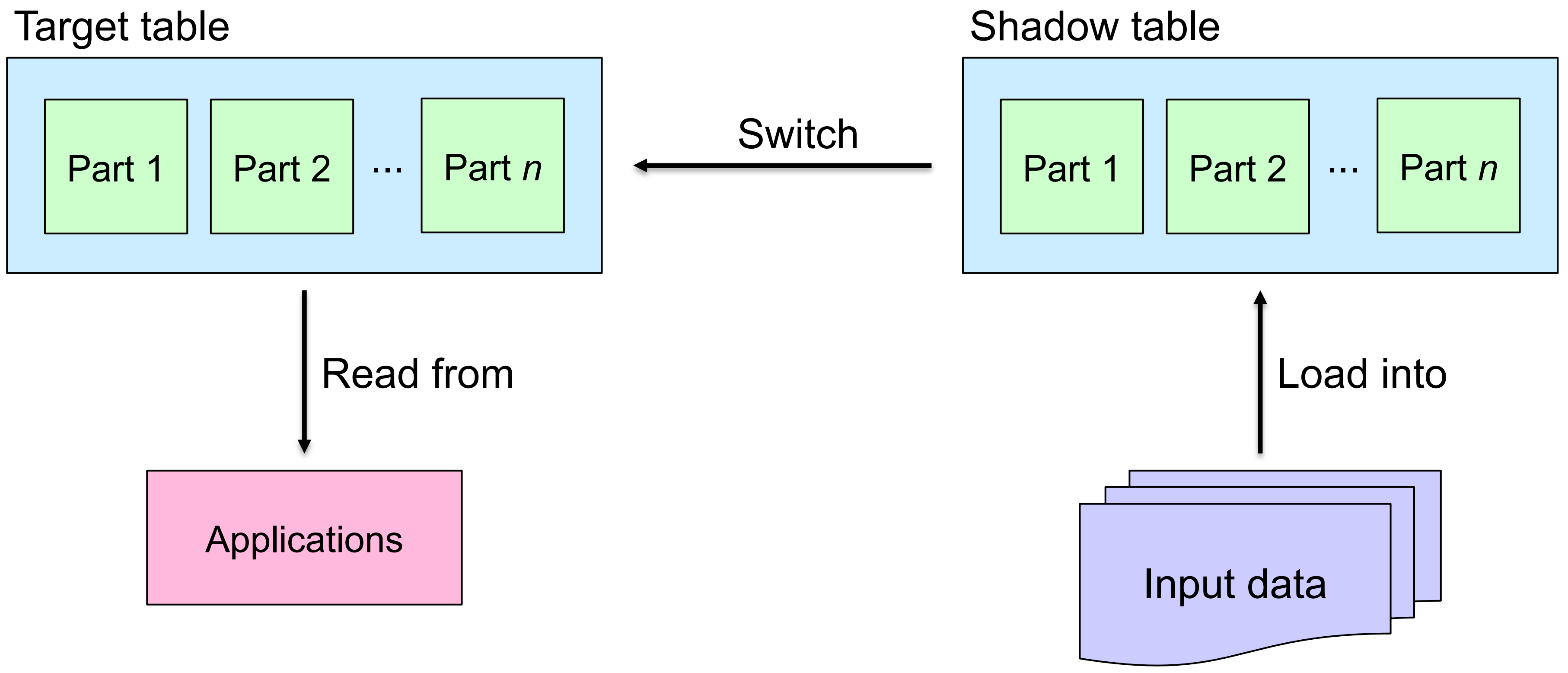
If, after reestablishing free space, data does not fit into A002 and A003 Abend04E - RC00E40318

New partition created for data overflow caused by reestablishing free space

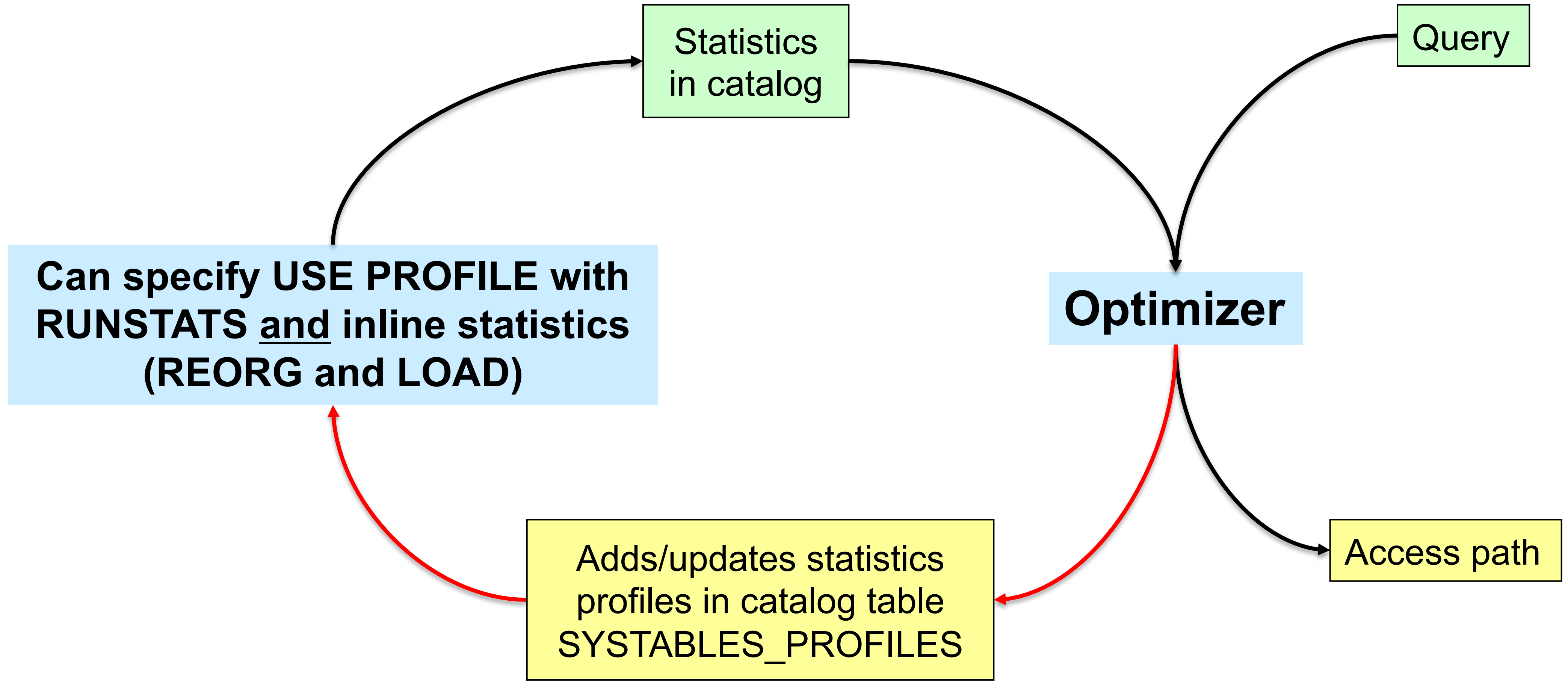
```
DSNU2910I -DB2A 266 03:46:49.81 DSNURAPT –
BASE PARTITION 5 IS CREATED
```

# Online LOAD REPLACE

- LOAD REPLACE SHRLEVEL REFERENCE (Db2 11: PI67793; Db2 12: PI69095)



# Automated statistics profiles


Statistics in catalog
Query
Can specify USE PROFILE with RUNSTATS and inline statistics (REORG and LOAD)
Optimizer
Adds/updates statistics profiles in catalog table SYSTABLES_PROFILES
Access path

**New with Db2 12**

# Thanks for your time.

Robert Catterall
rfcatter@us.ibm.com