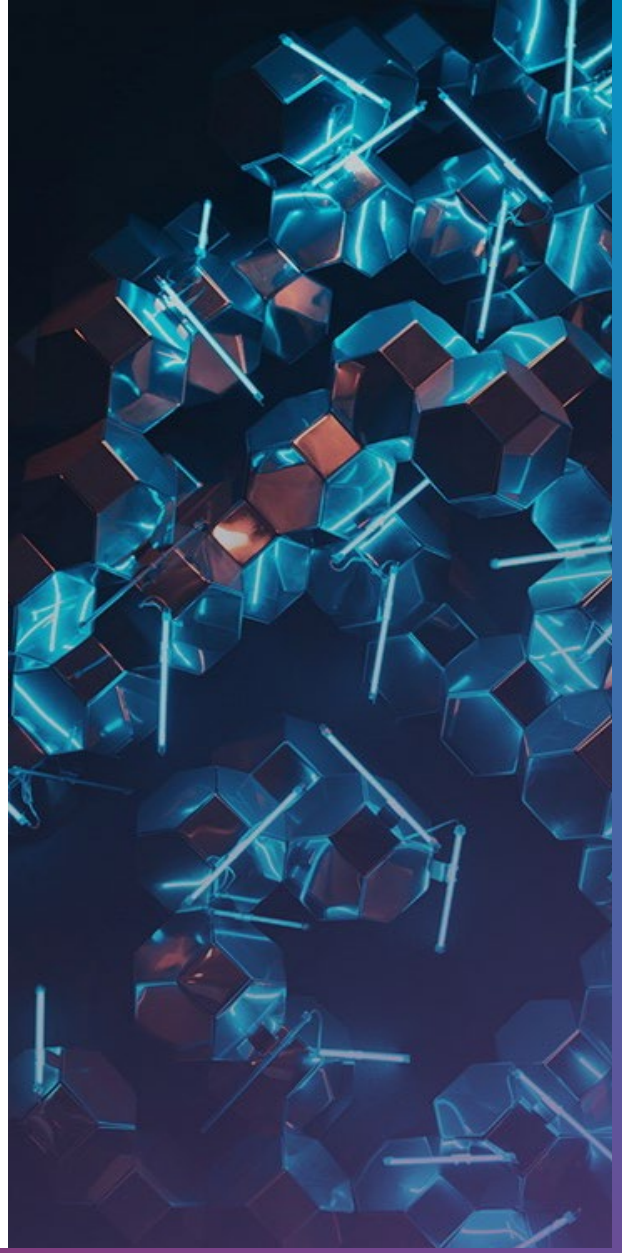
BROADCOM®

REORG

The Cornerstone of Housekeeping

Michal Bialecki @ PDUG, June 2026



Disclaimer

Certain information in this presentation may outline general product direction of Broadcom. This presentation shall not serve to (i) affect the rights and/or obligations of Broadcom or its licensees under any existing or future license agreement or services agreement relating to any Broadcom software product; or (ii) amend any product documentation of specification for any Broadcom software product. This presentation is based on current information and resource allocations as of June 11th, 2026 and is **subject to change or withdrawal by Broadcom at any time without notice. The development, release and timing of any features or functionality described in this presentation remain at Broadcom's sole discretion.**

Notwithstanding anything in this presentation to the contrary, upon the general availability of any future Broadcom product release referenced in this presentation, Broadcom may make such release available to new licensees in the form of a regularly scheduled major product release. Such release may be made available to licensees of the product who are active subscribers to Broadcom maintenance and support, on a when and if-available basis. The information in this presentation is not deemed to be incorporated into any contract.

THIS PRESENTATION IS FOR YOUR INFORMATIONAL PURPOSES ONLY. Broadcom assumes no responsibility for the accuracy or completeness of the information. TO THE EXTENT PERMITTED BY APPLICABLE LAW, BROADCOM PROVIDES THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTEIS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. In no event will Broadcom be liable for any loss or damage, direct or indirect, in connection with this presentation, including, without limitation, lost profits, lost investment, business interruption, goodwill, or lost data, even if Broadcom is expressly advised in advance of the possibility of such damages.

Row’s life in a Db2 table (1 – 5)

TABLE

id	first name	surname	address	position	salary (USD)
1	Emma	Johnson	123 Maple St, Springfield	Developer	52,000
2	Liam	Smith	456 Oak Ave, Rivertown	Business Manager	75,000
3	Olivia	Brown	789 Pine Rd, Lakeside	Staff	35,000
4	Noah	Jones	101 Elm St, Midway	Senior Developer	68,000
5	Ava	Garcia	202 Birch Blvd, Pleasantville	Developer	48,000
6	Elijah	Martinez	303 Cedar Ln, Hilltop	Staff	33,000
7	Sophia	Lee	404 Walnut Dr, Greenfield	Business Manager	79,000
8	Mason	Wilson	505 Cherry St, Brookside	Developer	54,000
9	Isabella	Anderson	606 Spruce Ct, Meadowbrook	Senior Developer	72,000
10	James	Thomas	707 Fir Pl, Crestwood	Staff	31,000
11	Mia	Taylor	808 Aspen Way, Sunnyvale	Developer	50,000
12	Benjamin	Moore	909 Poplar Rd, Westwood	Business Manager	74,000
13	Charlotte	Jackson	111 Maple St, Riverview	Staff	36,000
14	Lucas	White	222 Oak Ave, Oakdale	Developer	49,500
…					
10000	Michal	Bialecki	V Parku 8, Prague	Technical consultant	39,000

INDEX

id	first name	surname
1	Emma	Johnson
2	Liam	Smith
3	Olivia	Brown
4	Noah	Jones
5	Ava	Garcia
6	Elijah	Martinez
7	Sophia	Lee
8	Mason	Wilson
9	Isabella	Anderson
10	James	Thomas
11	Mia	Taylor
12	Benjamin	Moore
13	Charlotte	Jackson
14	Lucas	White
…		
10000	Michal	Bialecki

“Vanilla” table after initial LOAD or REORG, index entries point to row entries / ranges

Row's life in a Db2 table (2 – 5)

TABLE

id	first name	surname	address	position	salary (USD)
1	Emma	Johnson	123 Maple St, Springfield	Developer	52,000
2	Liam	Smith	456 Oak Ave, Rivertown	Business Manager	75,000
3	Olivia	Brown	789 Pine Rd, Lakeside	Staff	35,000
4	Noah	Jones	101 Elm St, Midway	Senior Developer	68,000
5	Ava	Garcia	202 Birch Blvd, Pleasantville	Developer	48,000
6	Elijah	Martinez	303 Cedar Ln, Hilltop	Staff	33,000
7	Sophia	Lee	404 Walnut Dr, Greenfield	Business Manager	79,000
8	Mason	Wilson	505 Cherry St, Brookside	Developer	54,000
9	Isabella	Anderson	606 Spruce Ct, Meadowbrook	Senior Developer	72,000
~~10~~	~~James~~	~~Thomas~~	~~707 Fir Pl, Crestwood~~	~~Staff~~	~~31,000~~
11	Mia	Taylor	808 Aspen Way, Sunnyvale	Developer	50,000
12	Benjamin	Moore	909 Poplar Rd, Westwood	Business Manager	74,000
13	Charlotte	Jackson	111 Maple St, Riverview	Staff	36,000
~~14~~	~~Lucas~~	~~White~~	~~222 Oak Ave, Oakdale~~	~~Developer~~	~~49,500~~
…					
10000	Michal	Bialecki	V Parku 8, Prague	Technical consultant	39,000

INDEX

id	first name	surname
1	Emma	Johnson
2	Liam	Smith
3	Olivia	Brown
4	Noah	Jones
5	Ava	Garcia
6	Elijah	Martinez
7	Sophia	Lee
8	Mason	Wilson
9	Isabella	Anderson
~~10~~	~~James~~	~~Thomas~~
11	Mia	Taylor
12	Benjamin	Moore
13	Charlotte	Jackson
~~14~~	~~Lucas~~	~~White~~
…		
10000	Michal	Bialecki

DELETE FROM TABLE …
WHERE …

Pseudo-deleted rows (pages) in index and table – "unusable/wasted" holes to be skipped during access

Row's life in a Db2 table (3 – 5)

TABLE

id	first name	surname	address	position	salary (USD)
1	Emma	Johnson	123 Maple St, Springfield	Developer	52,000
2	Liam	Smith	456 Oak Ave, Rivertown	Business Manager	75,000
3	Olivia	Brown	789 Pine Rd, Lakeside	Staff	35,000
4	Noah	Jones	101 Elm St, Midway	Senior Developer	68,000
5	Ava	Garcia	202 Birch Blvd, Pleasantville	Developer	48,000
6	Elijah	Martinez	303 Cedar Ln, Hilltop	Staff	33,000
7	Sophia	Lee	404 Walnut Dr, Greenfield	Business Manager	79,000
8	Mason	Wilson	505 Cherry St, Brookside	Developer	54,000
9	Isabella	Anderson	606 Spruce Ct, Meadowbrook	Senior Developer	72,000
~~10~~	~~James~~	~~Thomas~~	~~707 Fir Pl, Crestwood~~	~~Staff~~	~~31,000~~
11	Mia	Taylor	808 Aspen Way, Sunnyvale	Developer	50,000
12	Benjamin	Moore	909 Poplar Rd, Westwood	Business Manager	74,000
13	Charlotte	Jackson	111 Maple St, Riverview	Staff	36,000
~~14~~	~~Lucas~~	~~White~~	~~222 Oak Ave, Oakdale~~	~~Developer~~	~~49,500~~
…					
10000	Michal	Bialecki	V Parku 8, Prague	Technical consultant	39,000
15	**Amelia**	**Harris**	**333 Pine Rd, Fairview**	**Senior Developer**	**69,000**
16	**Henry**	**Martin**	**444 Elm St, Southport**	**Staff**	**34,000**

INDEX

id	first name	surname
1	Emma	Johnson
2	Liam	Smith
3	Olivia	Brown
4	Noah	Jones
5	Ava	Garcia
6	Elijah	Martinez
7	Sophia	Lee
8	Mason	Wilson
9	Isabella	Anderson
~~10~~	~~James~~	~~Thomas~~
11	Mia	Taylor
12	Benjamin	Moore
13	Charlotte	Jackson
~~14~~	~~Lucas~~	~~White~~
…		
10000	Michal	Bialecki
15	**Amelia**	**Harris**
16	**Henry**	**Martin**

INSERT INTO TABLE …

Out of sequence rows (pages) in index and table

Row's life in a Db2 table (4 – 5)

TABLE

	id	first name	surname	address	position	salary (USD)
	1	Emma	Johnson	123 Maple St, Springfield	Developer	52,000
	2	Liam	Smith	456 Oak Ave, Rivertown	Business Manager	75,000
	3	Olivia	Brown	789 Pine Rd, Lakeside	Staff	35,000
	4	Noah	Jones	101 Elm St, Midway	Senior Developer	68,000
	5	Ava	Garcia	202 Birch Blvd, Pleasantville	Developer	48,000
≠	~~6~~	~~Elijah~~	~~Martinez~~	~~303 Cedar Ln, Hilltop~~	Address of updated row	
	7	Sophia	Lee	404 Walnut Dr, Greenfield	Business Manager	79,000
	8	Mason	Wilson	505 Cherry St, Brookside	Developer	54,000
	9	Isabella	Anderson	606 Spruce Ct, Meadowbrook	Senior Developer	72,000
	~~10~~	~~James~~	~~Thomas~~	~~707 Fir Pl, Crestwood~~	~~Staff~~	~~31,000~~
	11	Mia	Taylor	808 Aspen Way, Sunnyvale	Developer	50,000
	12	Benjamin	Moore	909 Poplar Rd, Westwood	Business Manager	74,000
	13	Charlotte	Jackson	111 Maple St, Riverview	Staff	36,000
	~~14~~	~~Lucas~~	~~White~~	~~222 Oak Ave, Oakdale~~	~~Developer~~	~~49,500~~
	…					
	10000	Michal	Bialecki	V Parku 8, Prague	Technical consultant	39,000
	15	Amelia	Harris	333 Pine Rd, Fairview	Senior Developer	69,000
	16	Henry	Martin	444 Elm St, Southport	Staff	34,000
	6	**Elijah**	**Martinez**	**626 Mazda Dr, Meadowfield**	**Staff**	**33,000**

INDEX

id	first name	surname
1	Emma	Johnson
2	Liam	Smith
3	Olivia	Brown
4	Noah	Jones
5	Ava	Garcia
6	Elijah	Martinez
7	Sophia	Lee
8	Mason	Wilson
9	Isabella	Anderson
~~10~~	~~James~~	~~Thomas~~
11	Mia	Taylor
12	Benjamin	Moore
13	Charlotte	Jackson
~~14~~	~~Lucas~~	~~White~~
…		
10000	Michal	Bialecki
15	Amelia	Harris
16	Henry	Martin

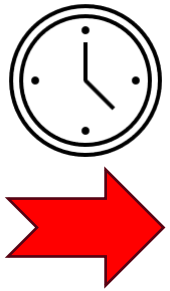
UPDATE <table> SET ADDRESS = 'new long address'

Non-in-place update creates an **indirect reference** row in the table when the inserted value occupies more space: the new row is an **overflow row** and the old row is a pointer ("indirect reference")

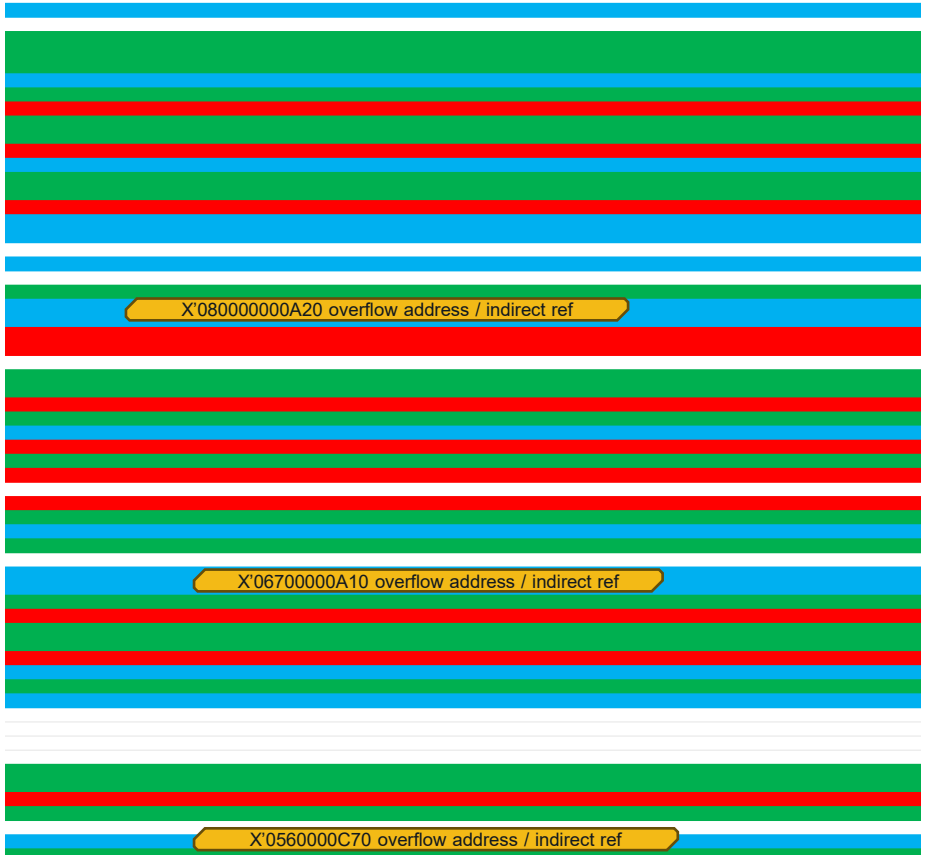
Index entry points to the original row

Row's life in a Db2 table (5 – 5)

id	first name	surname	address	position	salary (USD)
1	Emma	Johnson	123 Maple St, Springfield	Developer	52,000
2	Liam	Smith	456 Oak Ave, Rivertown	Business Manager	75,000
3	Olivia	Brown	789 Pine Rd, Lakeside	Staff	35,000
4	Noah	Jones	101 Elm St, Midway	Senior Developer	68,000
5	Ava	Garcia	202 Birch Blvd, Pleasantville	Developer	48,000
6	Elijah	Martinez	303 Cedar Ln, Hilltop	Staff	33,000
7	Sophia	Lee	404 Walnut Dr, Greenfield	Business Manager	79,000
8	Mason	Wilson	505 Cherry St, Brookside	Developer	54,000
9	Isabella	Anderson	606 Spruce Ct, Meadowbrook	Senior Developer	72,000
10	James	Thomas	707 Fir Pl, Crestwood	Staff	31,000
11	Mia	Taylor	808 Aspen Way, Sunnyvale	Developer	50,000
12	Benjamin	Moore	909 Poplar Rd, Westwood	Business Manager	74,000
13	Charlotte	Jackson	111 Maple St, Riverview	Staff	36,000
14	Lucas	White	222 Oak Ave, Oakdale	Developer	49,500
15	Amelia	Harris	333 Pine Rd, Fairview	Senior Developer	69,000
16	Henry	Martin	444 Elm St, Southport	Staff	34,000
17	Evelyn	Thompson	555 Birch Blvd, Northfield	Business Manager	77,000
18	Alexander	Garcia	666 Cedar Ln, Eastwood	Developer	53,000
19	Harper	Martinez	777 Walnut Dr, Lakewood	Senior Developer	70,000
20	Daniel	Robinson	888 Cherry St, Maplewood	Staff	30,000
…					
10000	Michal	Bialecki	V Parku 8, Prague	Db2 technical consultant	39,000

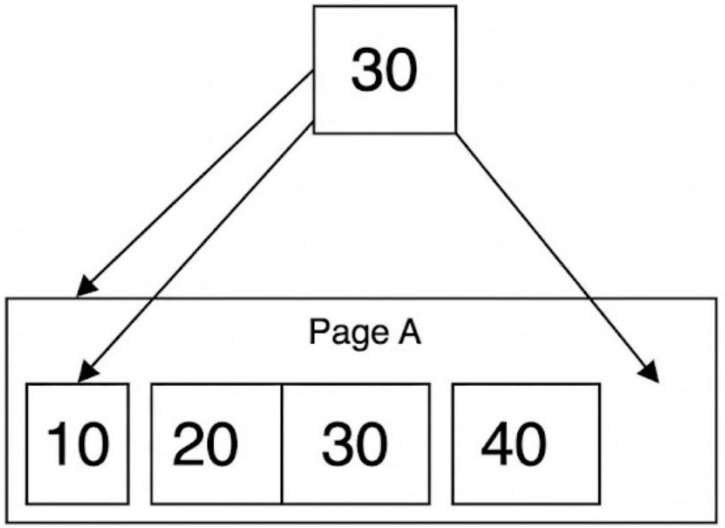


Performance is degraded due to hops – sync I/O when the page is not in BP



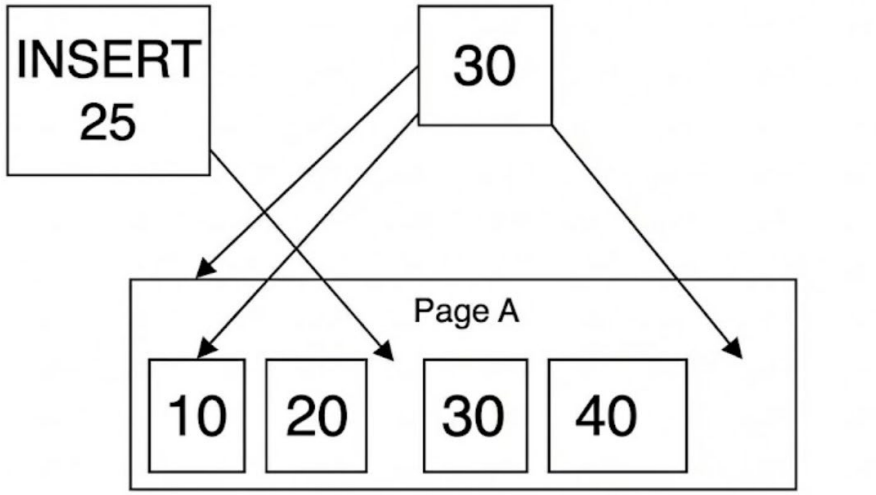
Index entry life in Db2 index (1 – 4)

- Imagine an index where each “leaf page" (a node in the B-Tree) can hold a maximum of 4 keys
- Assume Page A is completely full with values 10, 20, 30, and 40



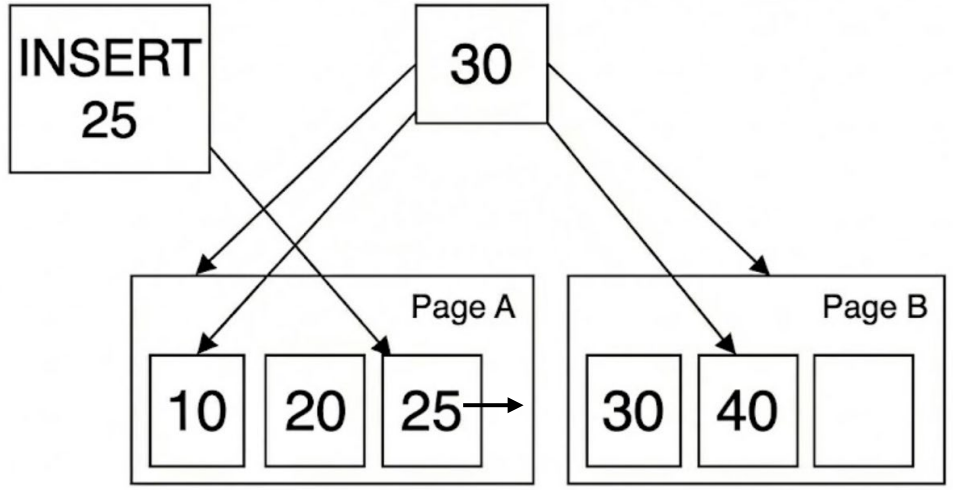
Index entry life in Db2 index (2 – 4)

- Next insert is attempted with value of 25
- B-Tree is traversed and determines that 25 belongs in Page A between 20 and 30
- Page A is already at its full capacity of 4 items
- This causes the **index page split**



Index entry life in Db2 index (3 – 4)

- Index Page Split:
  - It creates a new, empty page, Page B.
  - It takes all five values (10, 20, 25, 30, 40) and splits them roughly in half between the two pages
    - unless asymmetric split done, for monogamic insert patterns – e.g. timestamps
  - It updates the non-leaf-page node to point to both Page A and the new Page B
  - The result is two partially full pages, leaving room for future inserts.
  - Further splits would lead also to non-leaf index page split
    - can be chained splits up to root page – increases NLEVELS
  - During the index pages split of B-tree Db2 **latches the entire index** (next inserts or page split – queue)



Index entry life in Db2 index (4 – 4)

- How it maps to physical organization?
  - Split pages (page B) can be allocated ***anywhere***
  - Db2 tries to allocate it **near** (<16 pages), but may end up **far** distance (> 16 pages)
    - can be at the end of the index space, if no closer empty pages
  - Performance of index access degrades – Db2 has to "***jumps***" between logically neighboring pages that are now physically far away
  - Other conditions leading to **far reference** / jumps:
    - An index page is deleted and the distance between the new predecessor and successor pages is far

Performance is degraded due to hops when the page is not in BP as it needs a sync I/O

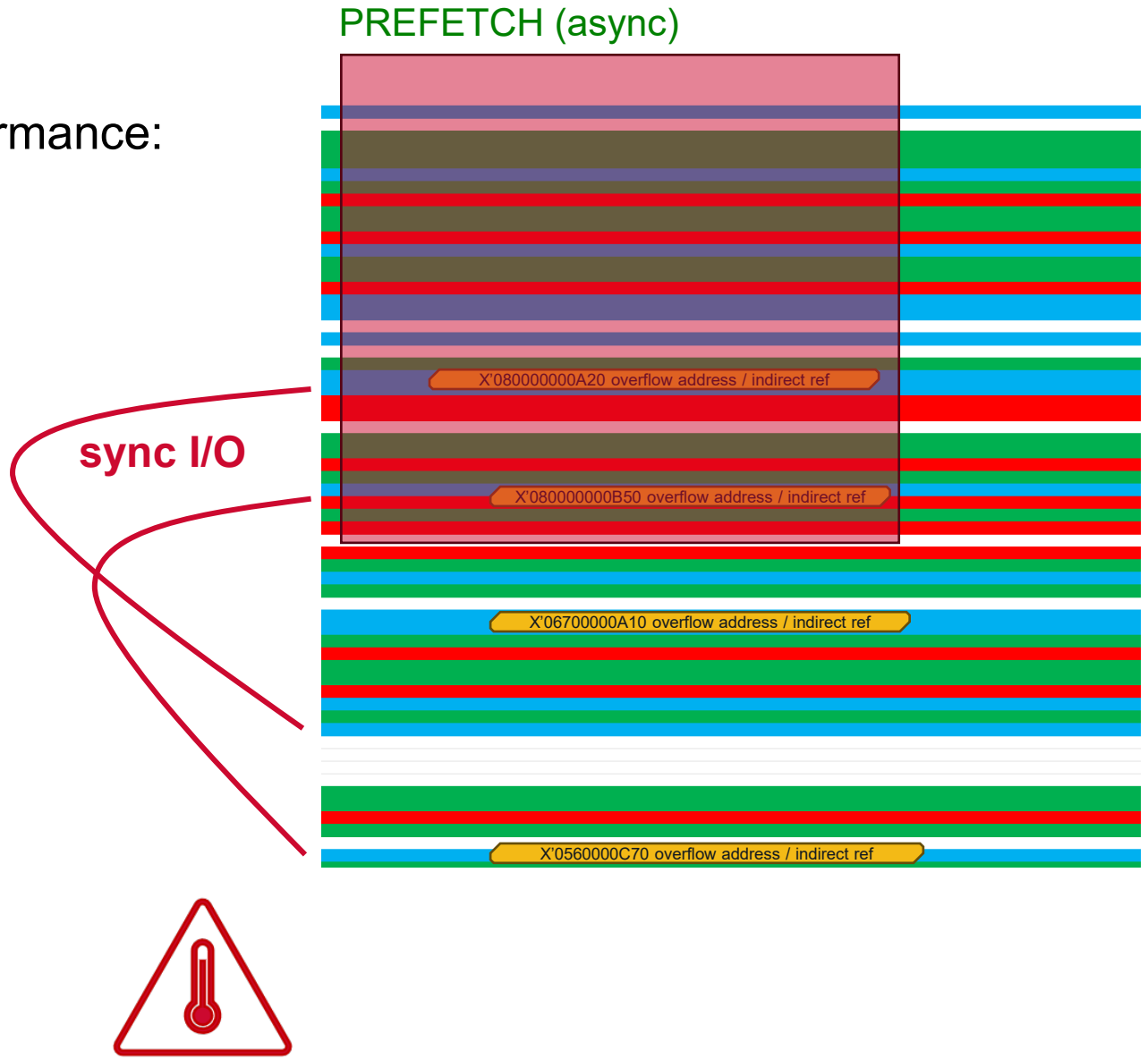
Access to table / index degrades over time due to (1 – 2)

- Un-clustered, out-of-sequence data
- Indirect references in **tablespaces**
  - Near indirect references (<16 pages distance from original page) SYSIBM.SYSTABLESPACESTATS.REORGNEARINDREF
  - Far indirect references (>16 pages distance from original page) SYSIBM.SYSTABLESPACESTATS. REORGFARINDREF
- Indirect references in **indexspaces**
  - SYSIBM.SYSINDEXSPACESTATS.REORGLEAFNEAR
  - SYSIBM.SYSINDEXSPACESTATS.REORGLEAFFAR
- Pseudo-empty rows / pages
- Partially filled partitions (excessive number of partitions)
- Datasets extents over many volumes
- Objects requiring Online Schema Changes (AREO*) and instantiation of PENDING changes

Access to table / index (2 – 2)

When we access table, we have degraded performance:

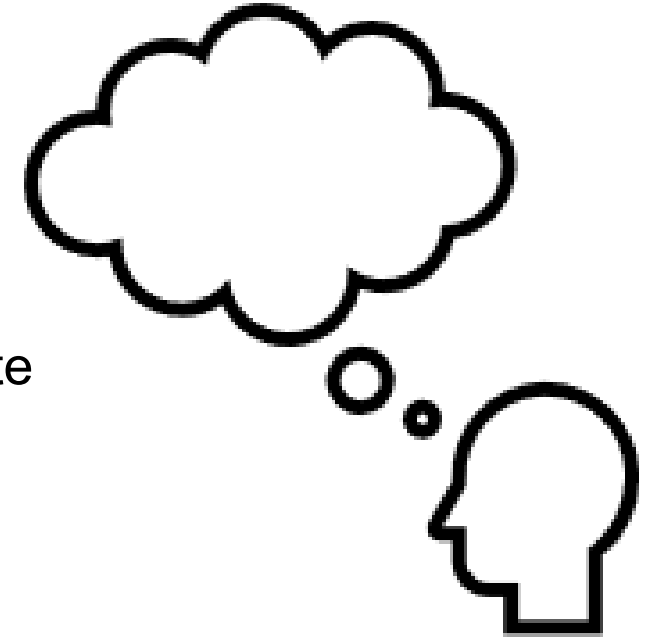
- more GETPAGES and I/Os due to
  - out of sequence access
  - fetching empty (pseudo-deleted) rows
- less efficient prefetch
  - for an index that has a CLUSTERRATIOF less than 80%, sequential prefetch is not used to access the data pages
- indirect references causing sync I/Os
  - highly likely that “far” indirect references won’t be part of async PREFETCH
  - sync IO leads to app waits for I/O to complete
- more locking, timeouts, deadlocks, latches
- space map contention, hot pages during insert



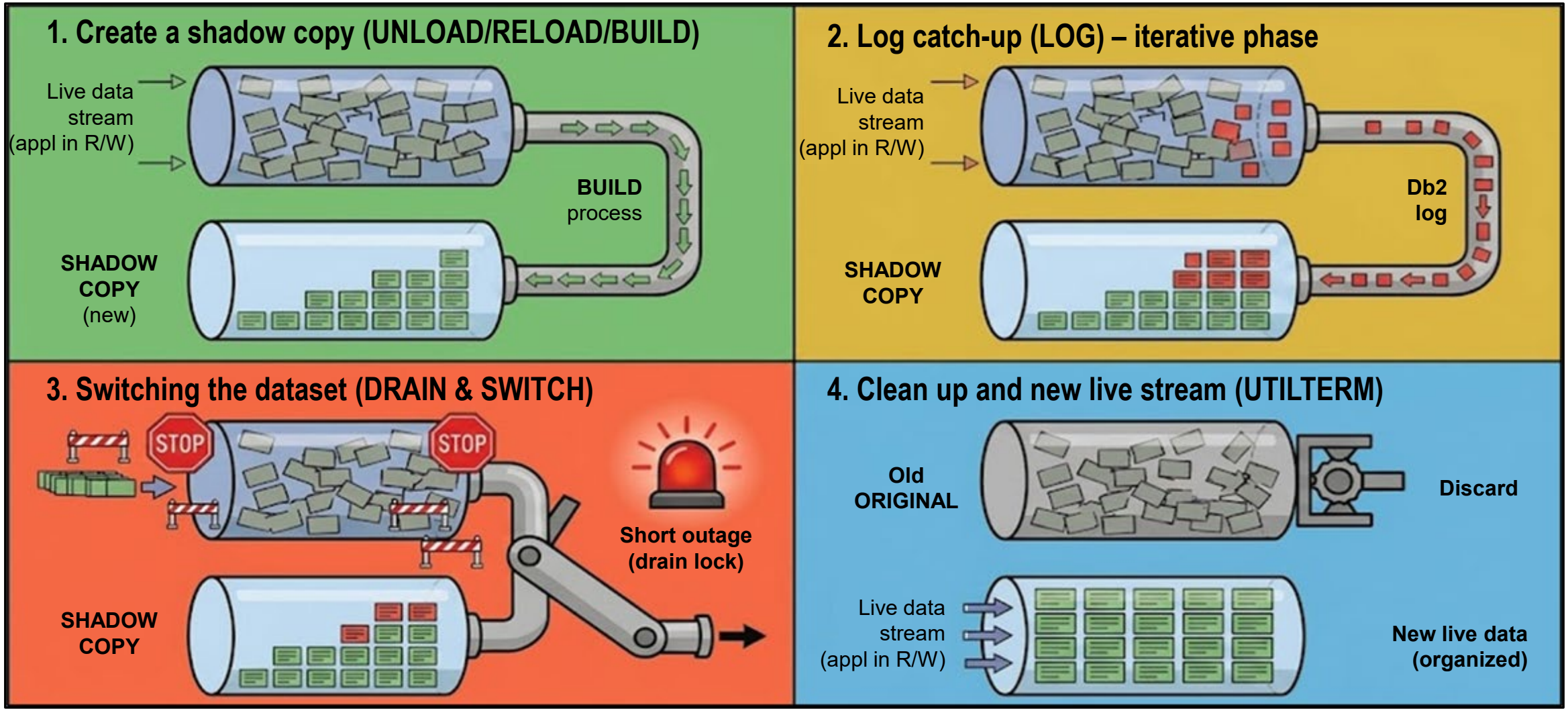
What now? REORG is a HERO.. Is it really?

After a while we need a REORG of Table or Index – this is obvious.. BUT!

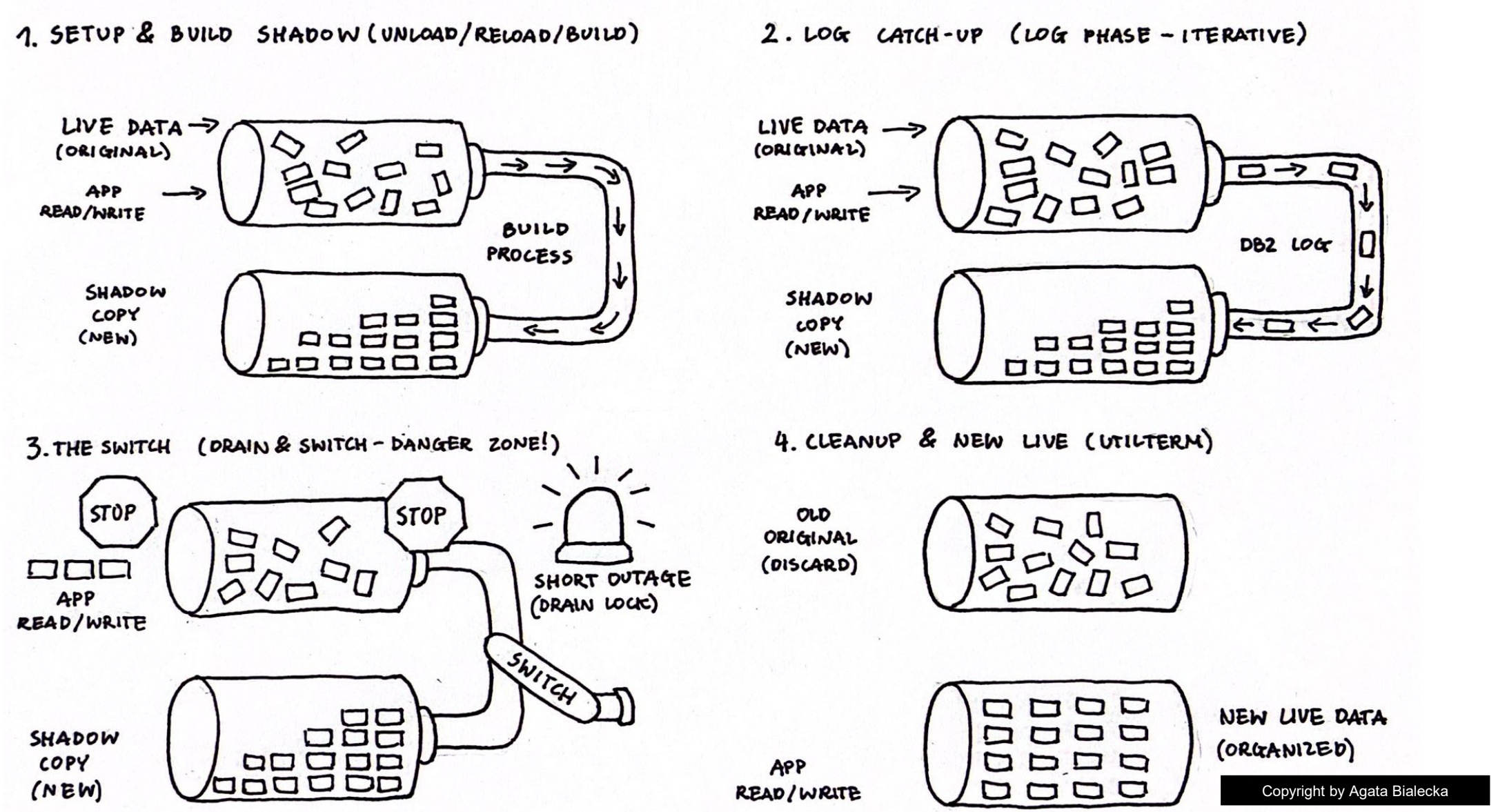
- Which options to use in my REORG?
- How to evaluate which objects need a REORG ?
- Shall I REORG part or REORG whole tablespace if partition(s) selected
- I would like to eliminate index REORG, when I already have the tablespace selected for REORG
- What about PBGs REORG, are they special ?
- How and when to schedule REORGs, how to build jobs ?
- Are all object the same importance ?
- How long they may take? Can all objects fit into maintenance window?
- What to do with XXL objects ? REORG may take hours / days to complete
  - Shall I just REORG INDEX?
- How about RUNSTATS – together with REORG or outside of REORG?
- Shall I run RUNSTATS with REORG INDEX?



REORG SHRLEVEL CHANGE Phases



REORG SHRLEVEL CHANGE Phases – Agata's version



Which options to use in my REORG? (1 – 5)

- Use SHRLEVEL CHANGE to avoid (minimize) application outage
- Avoid transaction timeouts during REORG:
  - Set MAXRO and DRAIN_WAIT as follows:
    - (MAXRO + DRAIN_WAIT) < (IRLMRWT – few* seconds) *few* = 3-5 seconds*
      - let REORG timeout first before your application timeouts (eg online transaction) if there is any collision
  - LASTLOG NO (APAR PH4652 (2022)) – watches for NO log records to be applied in the final log iteration or abandon it if criteria is not met
    - high RETRY value (max of 255) is recommended to lessen the impact of the repeating "break in" attempts
- Use DRAIN ALL – drain READERS and WRITERS at same time
  - Otherwise, if DRAIN WRITERS, DRAIN happens twice (READERS and WRITERS separately)
- RETRY / RETRY_DELAY
  - If the Drain fails, how many times should REORG try again?

Which options to use in my REORG? (2 – 5)

How about BIG (HUGE) tables

- Run REORG with MAXRO DEFER, schedule the REORG outside the REORG window, and ALTER UTILITY REORG MAXRO n inside REORG window in quiet time
  - How to determine when quiet time is:
    - Based on history e.g. using Log Analyzer™ for Db2 for z/OS / Quiet Point Analysis
    - `-DISPLAY BLOCKERS` command
    - `SELECT .. FROM TABLE (SYSIBMADM.BLOCKING_THREADS('*'))`

Which options to use in my REORG? (3 – 5)

- Use inline STATISTICS with REORG TABLESPACE
  - creates “island of stability” for access path selection –
    - perfectly organized data and statistics reflecting it
    - you can go back to it, with next REORGs/LOADs
- Don’t use inline STATISTICS (RUNSTATS) with REORG INDEX
  - creates “time drift” between access path statistics for table vs. index
  - may cause also conflicting statistics between index and table statistics
- Avoid running standalone RUNSTATS
  - It will update statistics to reflect un-clustered data in table, not the best access path may be chosen then
  - It will lose “island of stability” –
    - impossible to repeat scenario / combination for predictive access path selection to revert too
    - you may “accidentally” get a good access path but very fragile
      - minimal costs difference can flip it to good / bad, that you never can go back to
  - Run only when absolutely necessary
    - Eg: can’t afford to run with REORG in mid day, and you need to correct access path somehow – lucky hit?

Which options to use in my REORG? (4 – 5)

- NOSYSREC – uses the output of sorting as the input to reload but does not use an unload data set for this process
  - Omitting the unload data set can improve performance

- SORTDEVT – if you do not specify it, Db2 will assume few things, on example:

```
DSNU397I    040 15:52:18.46 DSNURPIB - NUMBER OF TASKS CONSTRAINED BY MISSING SORTDEVT TO 5
DSNU397I    040 15:52:18.46 DSNURPIB - NUMBER OF TASKS CONSTRAINED BY CPUS TO 5
```

  – Instead of “running at full speed”:

```
DSNU395I    040 15:36:01.56 DSNURPRD - INDEXES WILL BE BUILT IN PARALLEL, NUMBER OF TASKS = 8
DSNU397I    040 15:36:01.56 DSNURPRD - NUMBER OF TASKS CONSTRAINED BY CPUS TO 24
```

  For dynamic allocation of sort work data sets, SORTDEVT must be specified

Which options to use in my REORG? (5 – 5)

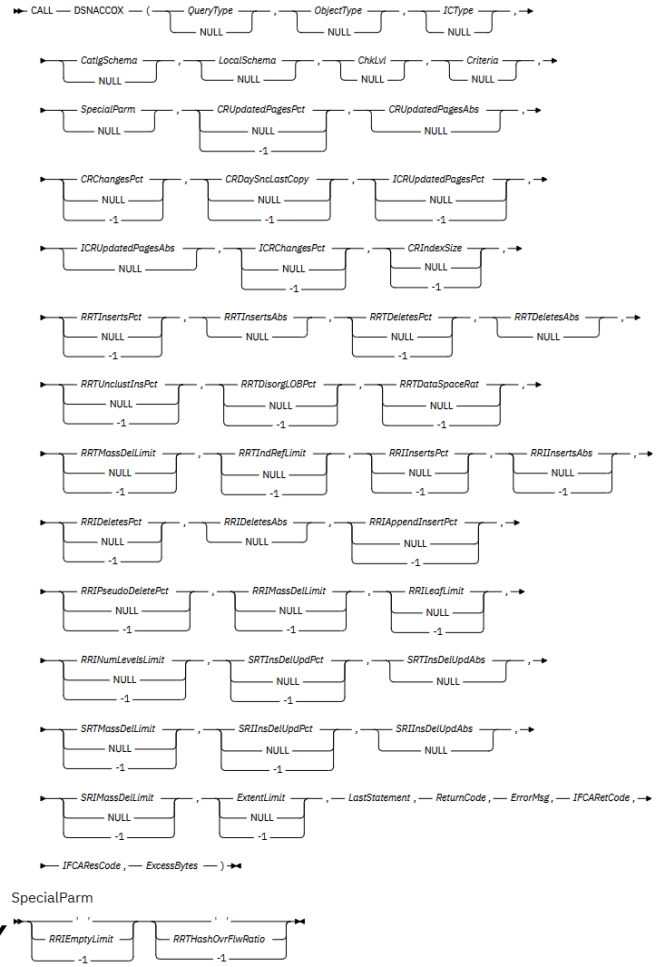
- Set zParm `REORG_DROP_PBG_PARTS` to `DISABLE`
  - otherwise, if partition is dropped object is not recoverable prior to REORG point time
  - use explicitly DROP_PART parm in REORG if really needed
- For inline image copies created during REORG
  - Don't use `STACK YES` image copies too wide and only when you can automatically generate jobs for RECOVER objects without colliding with other jobs accessing same TAPE (VTS is also seen as tape) - objects are accessed sequentially when STACKed
    - Vendor tools can help in creating "intelligent" recovery, for all datasets STACKed on one volume: example Broadcom RECOVERY ANALYZER, global option
      - **Maximum # of tapes** - number of volume **serials** allocated to a single Recovery job.
        A value of 1 avoids contention when different jobs try to access the same tape volume (serial).
  - specify a template name for `COPYDDN` or `RECOVERYDDN` that uses the &PA. or &PART. variable
    - REORG allocates as many copy data sets as the number of partitions that are being reorganized
    - otherwise one HUGE IC will be created for all PARTITIONS

How to evaluate which objects need a REORG? (1 – 3)

Few options:

- IBM provides sample stored procedure DSNACCOX:
  - hard to use, too many input variables to be set, no clear guidance on recommended values
  - not much logic to control for different conditions / different types of objects
  - “partial solution” - requires programing knowledge to transform selected object into REORG statements / JCL / automation
- Vendor solutions, eg Database Analyzer™ for Db2 for z/OS – out of the box solution:
  - updated in regular maintenance stream (PTFs) for any SQL changes and new features
  - extremely flexible in addressing custom needs and selection criteria, and prioritization
  - generates not only list of REORG candidates but from A..Z ready JCLs
  - can be to be plugged into existing automation
  - can be “set and forget” without harm – once set correctly will simply do the work
- In-house SQLs based on RTS: DSNACCOX / DSNACCOR
  - requires manual maintenance - regular SQL review and programming skills
  - SQL has no info about REOR* statuses
  - often set and forgot (DSNACCOR SQL is obsolete)
- “old” in-house SQL based on Statistics (non-RTS)
- using deprecated REORG options OFFPOSLIMIT / INDREFLIMIT / REPORTONLY
  - to decide on REORG you first need to execute RUNSTATS (!)

https://www.ibm.com/docs/en/db2-for-zos/13.0.0?topic=db2-dsnaccox



How to evaluate which objects need a REORG? (2 – 3)

DSNACCOX thresholds for REORG **Tablespace**

```
(((QueryType='REORG' OR QueryType='ALL') AND
  (ObjectType='TS' OR ObjectType='ALL')) AND
 (Object is not in Persistent Read Only (PRO) status) AND
( REORGLASTTIME IS NULL OR
 (NACTIVE IS NULL OR NACTIVE > 5) AND
 ((((REORGINSERTS×100)/TOTALROWS>RRTInsertsPct)1 AND
   REORGINSERTS>RRTInsertsAbs) OR
  (((REORGDELETES×100)/TOTALROWS>RRTDeletesPct) AND
   REORGDELETES>RRTDeletesAbs) OR
   ((REORGCLUSTERSENS > 0 OR
    (REORGCLUSTERSENS IS NULL)) AND
     (DRIVETYPE = 'SSD' AND
      (REORGUNCLUSTINS×100)/TOTALROWS >
                  (RRTUnclustInsPct * SSDMultiplier5))) OR
    (DRIVETYPE = 'HDD' AND
     (REORGUNCLUSTINS×100)/TOTALROWS>
                  RRTUnclustInsPct))) OR
   (REORGDISORGLOB×100)/TOTALROWS>RRTDisorgLOBPct OR
   (Not HASH organized and
    (SPACE > RRTDataSpaceRat × (DATASIZE/1024))) OR
   ((REORGNEARINDREF+REORGFARINDREF)×100)/TOTALROWS>
                                      RRTIndRefLimit OR
   REORGMASSDELETE>RRTMassDelLimit OR
   EXTENTS>ExtentLimit4)))) OR
 ((QueryType='REORG' OR QueryType='ALL') AND
  objectType='ALL'2 AND
  overflow index for hash access is used3 AND
  (overflow index TOTALENTRY ×100) / TOTALROWS > RRThashOvrFlwRatio)) OR
 ((QueryType='RESTRICT' OR QueryType='ALL' OR QueryType='REORG') AND
  (ObjectType='TS' OR ObjectType='ALL') AND
  The table space is in advisory of informational reorg pending status)
```

https://www.ibm.com/docs/en/db2-for-zos/13.0.0?topic=db2-dsnaccox

- Thresholds + comments
  - **RRTIndrefLimit = 5% (datasharing), 10% (non-DS)**
    - Far and near indirect references (overflow records)
  - **Note**: For small tables all percentage thresholds can be reached too quick, I wish DSNACCOX would consider also TOTALROWS as input parm, to control:
    - RRTUnclustInsPct = 10% default
      - may reach too quick for small tables
    - RRTDeletePct = 25% default
      - may reach too quick for small tables if not guarded by RRTDeletesAbs (0 by default)
    - RRTInsertPct, RRTInsertsAbs
      - may reach too quick for small tables, don't set – by default is disabled
  - RRTDisorgLOBPct = 50 %
    - disorganized LOB chunks
  - RRTMassDelLimit = 0
  - ExtentLimit = 254
    - SMS managed VSAM can have up to 7257 extents, if the extent constraint removal parameter in the DataClass is set to Y

How to evaluate which objects need a REORG? (3 – 3)

DSNACCOX thresholds for REORG **Index**

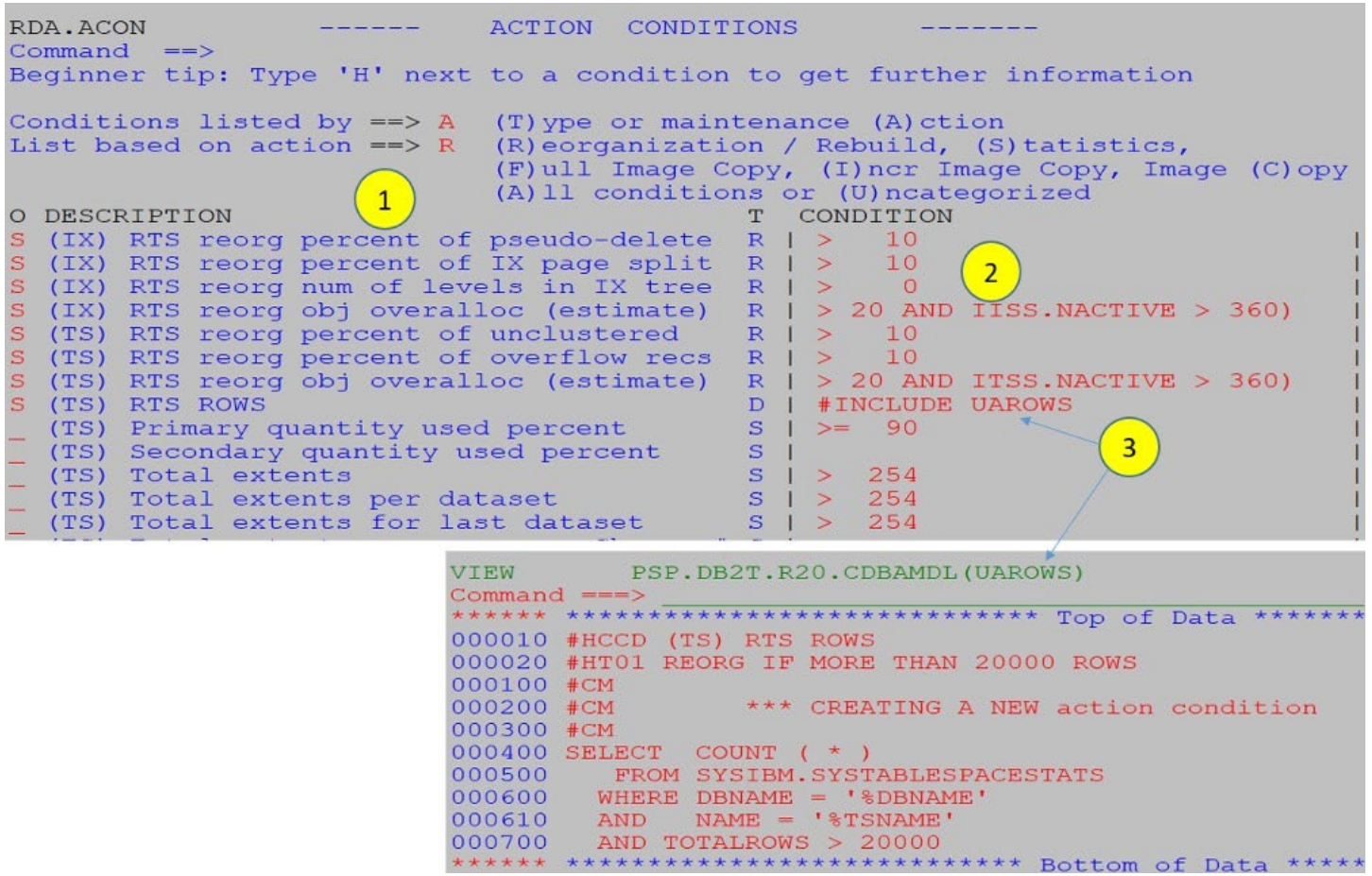
```
(((QueryType='REORG' OR QueryType='ALL') AND
  (ObjectType='IX' OR ObjectType='ALL') AND
  (REORGLASTTIME IS NULL AND REBUILDLASTTIME IS NULL) OR
  (NACTIVE IS NULL OR NACTIVE > 5) AND
  ((((REORGINSERTS×100)/TOTALENTRIES>RRIInsertsPct) AND
    REORGINSERTS>RRIInsertsAbs)1 OR
   (((REORGDELETES×100)/TOTALENTRIES>RRIDeletesPct) AND
    REORGDELETES>RRIDeletesAbs) OR
   (REORGAPPENDINSERT×100)/TOTALENTRIES>RRIAppendInsertPct OR
   (REORGPSEUDODELETES×100)/TOTALENTRIES>RRIPseudoDeletePct2 OR
   REORGMASSDELETE>RRIMassDelLimit OR
   (REORGLEAFFAR×100)/NACTIVE>RRILeafLimit OR
   REORGNUMLEVELS>RRINumLevelsLimit OR (NPAGES>5 AND
(NPAGES*100)/NLEAF>RRIEmptyLimit) OR
   EXTENTS>ExtentLimit)) OR
 ((QueryType='RESTRICT' OR QueryType='ALL' OR QueryType='REORG') AND
  (ObjectType='IX' OR ObjectType='ALL') AND
  An index is in advisory-REBUILD-pending stats (ARBDP)
  or an index is in REBUILD-pending empty status (RBDPM)))
```

https://www.ibm.com/docs/en/db2-for-zos/13.0.0?topic=db2-dsnaccox

- Thresholds + comments
  - **RRILeafLimit = 10 %**
    - Far indirect references (ratio of splits / active pages)
      - due to index splits or deleted empty pages
  - RRIEmptyLimit = 5%
    - ratio of pseudo-empty pages (non usable free pages) to the total number of leaf pages
  - RRINumLevelsLimit = 0 – levels increased since last REORG
  - **Note**: Percentage thresholds may trigger too quick, I wish DSNACCOX would consider also TOTALENTRIES as parm for:
    - RRIPseudoDeletePct = 5 %
      - Only marked as deleted, non-usable free index entries
    - RRIAppendInsertPct (20 %) RRIDeletesPct (30 %)
      - may reach too quick for small tables
    - RRIInsertsPct (-1), RRIInsertsAbs (0)
      - may reach too quick for small tables, don't set – by default is disabled
  - RRIMassDelLimit = 0
  - ExtentLimit = 254
    - SMS managed VSAM can have up to 7257 extents, if the extent constraint removal parameter in the DataClass is set to Y

Example of vendor solution – Database Analyzer (1 – 3)

1. The conditions list has been filtered to display only those related to reorganizations
2. Dozens of predefined RTS conditions are available (for example, for index page splits or an increase in the number of index levels that cause additional I/O processing)
3. You can also define your own custom conditions in SQL, based on your site-specific requirements, e.g. those who have more rows than 20000



Example of vendor solution – Database Analyzer (2 – 3)

- REORG using RTS conditions
  - Can simply select all i.e. DSNACCOX with the default thresholds
  - Or pick and choose conditions and customise the thresholds

```
Conditions listed by ==> A  (T)ype or maintenance (A)ction
List based on action ==> A  (R)eorganization / Rebuild, (S)
                            (F)ull Image Copy, (I)ncr Image Copy, Image (C)opy
                            (A)ll conditions or (U)ncategorized
O DESCRIPTION                                        T  CONDITION
S (TS) Status:AREO*,REORP                            R  ;
S (IX) RTS DSNACCOX too many extents                 X  >   254
S (IX) RTS perc app_ins since last reorg             R  >    10
S (IX) RTS reorg percent of pseudo-delete            R  >    10
S (IX) RTS reorg percent of IX page split            R  >    10
S (IX) RTS reorg num of levels in IX tree            R  >     0
S (TS) RTS DSNACCOX too many extents                 X  >   254
S (TS) RTS load (and reorg) has not run              X  IS NULL)
S (TS) RTS reorg percent of unclustered              R  >    10
S (TS) RTS reorg percent of overflow recs            R  >    10
_ (TS) Average size of a LOB in bytes                D
_ (TS) Ratio of organization in the LOB              D
_ (TS) Percent of freespace in the LOB               D
_ (TS) Tablespace type (I,K,L,O, or ' ')             D
_ (TS) Check pending status                          D
```

Example of vendor solution – Database Analyzer (3 – 3)

JCL generation on selected criteria / thresholds

```
EDIT                  DIV.CNTL(MTEACT01) - 01.01                Columns 00001 00072
Command ===>                                                     Scroll ===> CSR
****** ***************************** Top of Data ********************************
000001 //MTEACT01 JOB CLASS=A,MSGCLASS=X
000002 //* RO        PSP.DB2T.R20.CDBAMDL()
000003 //UTIL0001  EXEC PGM=DSNUTILB,REGION=4096K,PARM='D121'
000004 //STEPLIB   DD DISP=SHR,DSN=EOSDB2.DB2C10.SDSNLOAD
000005 //          DD DISP=SHR,DSN=D120.PRIVATE.SDSNEXIT
000006 //SYSPRINT DD SYSOUT=*
000007 //UTPRINT   DD SYSOUT=*
000008 //SYSUT1    DD UNIT=SYSDA,DISP=(,DELETE),
000009 //             SPACE=(CYL,(5,20))
000010 //SORTOUT   DD UNIT=SYSDA,DISP=(,DELETE),
000011 //             SPACE=(CYL,(5,20))
000012 //SYSDISC   DD DISP=SHR,DSN=NULLFILE
000013 //SYSPUNCH DD DISP=SHR,DSN=NULLFILE
000014 //SYSIN   DD   *
000015 LISTDEF L0000001
000016   INCLUDE TABLESPACE END8D12A.END8S12A PARTLEVEL 0001
000017   INCLUDE TABLESPACE END8D12A.END8S12D PARTLEVEL 0001
000018   INCLUDE TABLESPACE END8D12A.END8S12E PARTLEVEL 0001
000019   INCLUDE TABLESPACE END8D12A.END8S12E PARTLEVEL 0002
000020   INCLUDE TABLESPACE END8D12A.END8S12E PARTLEVEL 0003
000021   INCLUDE TABLESPACE END8D12A.END8S12E PARTLEVEL 0005
000022   TEMPLATE CDATA1 DSN '&USERID..&DB..&TS..P&PA(2)..D&DT(3)..C&TI(1,5).'
000023     UNIT VTAPE RETPD 15 VOLCNT(5) DISP(NEW,CATLG,DELETE)
000024   TEMPLATE CDATA2 DSN '&USERID..&DB..&TS..P&PA(2)..D&DT(3)..C&TI(1,5).'
```

```
T01) - 01.01                Columns 00001 00072
                             Scroll ===> CSR
TLG,DELETE)
L
RID..&DB..&TS..P&PA(2)..D&DT(3)..T&TI(1,5).'
TLG,DELETE)
L
D..&DB..&TS..P&PA(2)..D&DT(3)..T&TI(1,5).'
TLG,DELETE)
L
000033   REORG TABLESPACE LIST L0000001
000034         COPYDDN(CDATA2)
000035         UNLDDN(UDATA1)
000036         STATISTICS TABLE(ALL) INDEX(ALL)
000037         SORTDEVT SYSDA  SORTNUM 4
000038         UNLOAD CONTINUE
000039         SHRLEVEL CHANGE
000040         DEADLINE CURRENT TIMESTAMP + 1 HOUR
000041         MAXRO 240
000042         LONGLOG
000043         DRAIN DELAY 1200
000044         DRAIN_WAIT 30 RETRY 4 RETRY_DELAY 10
****** ***************************** Bottom of Data ****************************
```

The best REORG is no REORG! (1 – 2)

- Build a scoring table and why REORG was triggered
  - Maybe DDL change is the solution?
- If REORG TABLESPACE was due to
  - indirect references caused by inserts/updates into Table –
    - increase PCTFREE
    - increase PCTFREE FOR UPDATE (or use AUTO (-1))
  - number of un-clustered insert
    - increase SEGSIZE to provide better chance of maintaining clustering
- If REORG INDEX was due to
  - Index page splits and indirect references
    - Consider **larger index page size** to reduce index page splits
      - would need another BufferPool (and increased size of BufferPool)
  - pseudo-deleted index entries and pseudo-empty pages :
    - Increase ZPARM INDEX_CLEANUP_THREADS = 10 (default) system tasks

The best REORG is no REORG! (2 – 2)

- Merge candidates for REORG INDEX with candidates for REORG TABLESPACE
- If significant % of partitions is selected, it makes sense to REORG whole Tablespace
- If PBG part is selected or Tablespace with NPIs – non partitioning indexes, it make sense to REORG whole Tablespace - NPIs are REORGed as whole objects, not on index partition level

REORG sometimes can’t be avoided – sooner or later ..

- **REORG-pending (REORP) restrictive status**
  - Indicates that the object must be reorganized to apply definition changes before the data is accessible

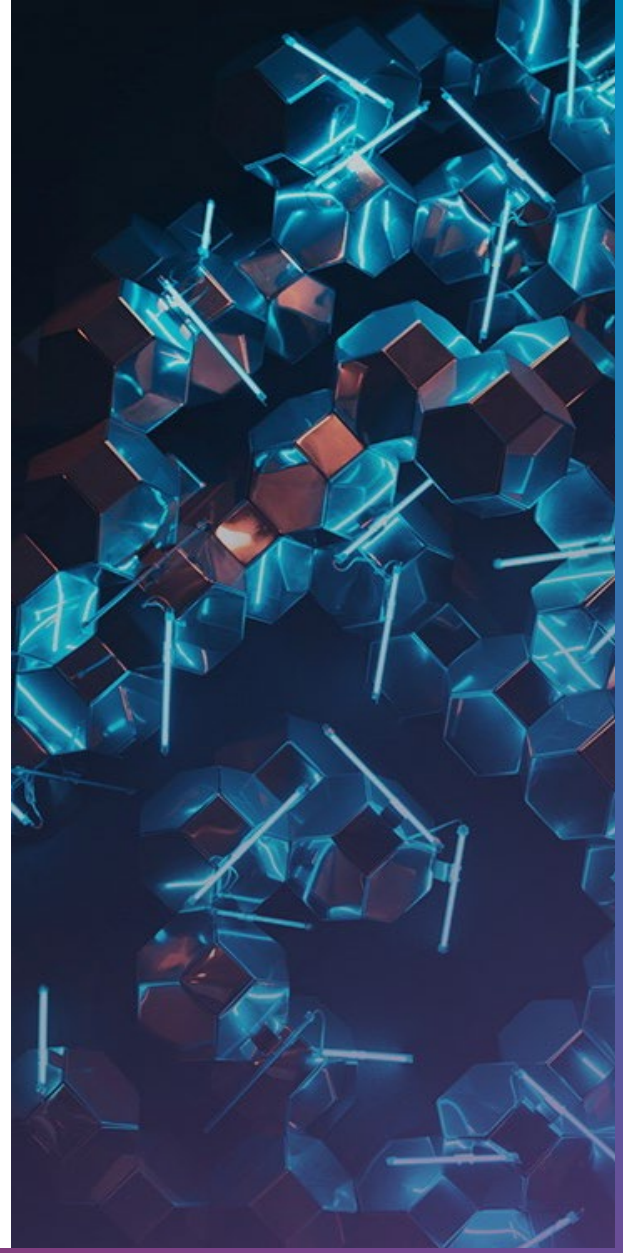
- **REORG-pending (AREO*) advisory status**
  - For **schema changes** applied immediately to table spaces
  - Indicates that the object needs to be reorganized for optimal performance.
  - The affected objects are not restricted and can be accessed by both readers and writers.

- **REORG-pending (AREOR) advisory status:**
  - Indicates that the object should be reorganized to materialize **pending definition** changes.
  - Changes are registered in the Db2 Catalog (SYSIBM.SYSPENDINGDDL)
  - The affected objects are not restricted and can be accessed by both readers and writers

Takeaways for Db2 Reorg

- **REORG can’t be 100% avoided but can be limited**
  - Proactively minimize maintenance by performing trend analysis, tuning your DDL (like PCTFREE or SEGSIZE), and enabling automatic index clean-up
- **Evaluate Intelligently:**
  - Stop guessing and use data-driven evaluations - like IBM's DSNACCOX formulas or **automated** vendor solutions - to target only the objects that actually need a REORG
- **Optimize for Zero Downtime:**
  - Prevent transaction timeouts and outages by utilizing SHRLEVEL CHANGE, specifying DRAIN ALL, and tightly pairing your MAXRO + DRAIN_WAIT < IRLMRWT,
  - for big problematic, intensively utilized tables use additionally MAXRO DEFER or/and LASTLOG NO
- **Defend Your "Island of Stability":**
  - Always collect inline statistics during tablespace REORGs, but strictly avoid them for index REORGs and eliminate ad-hoc RUNSTATS
- **REORG is only half the battle** – there is no point in REORG without REBIND.
  - **As a general rule - perform a "safe" REBIND** (APREUSE or/and APCOMPARE) to ensure the optimizer leverages your newly organized data without risking access path regressions,
  - if you **TRULY can evaluate** new access path – do REBIND without APREUSE / APCOMPARE to allow new access path choices that came in maintenance stream / new versions

Appendix



Appendix: REORG Restrictive / Advisory States (1 – 5)

State	Reason	Resolution
REORP REORG pending Restrictive **No** Read/Write Access	A Db2 table space, partition, or index is placed in a **REORG-pending (REORP) restrictive** status for several reasons related to schema alterations and point-in-time recoveries: • Recovering a table space to a **point in time before the materialization of pending definition changes** (for most types of pending changes) • Adding or altering partitions and **limit keys for partitioned (non-UTS)** table spaces with index-controlled partitioning - immediate definition change • **Adding a column that requires generated values** and cannot use the same default for every existing row will place the table space in a restrictive REORP status. Examples include ROWID, identity, row change timestamp, row-begin, row-end, or transaction-start-ID • **ALTER LOB** column with the INLINE LENGTH option to reduce the maximum length	REORG TABLESPACE (or LOAD REPLACE)

Appendix: REORG Restrictive / Advisory States (2 – 5)

State	Reason	Resolution
AREO* Advisory Reorg Full Read/Write Access	Set after immediate schema change that does not require an outage, but where a subsequent reorganization is recommended for optimal performance – after REORG rows can eventually be reloaded and converted to the new scheme definition: • **Adding columns:** Adding a new column to a table provided the new column doesn't require a generated value (like a ROWID or Identity column, which would trigger a restrictive REORP status) • **Modifying column attributes:** Altering a table to change a column's data type or to increase a column's length	REORG TABLESPACE / INDEX

Appendix: REORG Restrictive / Advisory States (3 – 5)

State	Reason	Resolution
AREOR **Advisory Reorg pending definition changes** Full Read/Write Access	• ALTER TABLESPACE .. **:** • **MAXPARTITIONS –** converting single-table simple or segmented (non-UTS) table spaces to PBG table spaces • **SEGSIZE eg** table space is being converted from a partitioned (non-UTS) table space to a partition-by-range (PBR) table space • **BUFFERPOOL (**different page size) • **DSSIZE** for APN (absolute page numbering) • **DSSIZE** for RPN (relative) to smaller size • **MEMBER CLUSTER** • **MOVE TABLE** • **PAGENUM**	REORG SHRLEVEL CHANGE or REFERENCE **(not NONE !)**

Appendix: REORG Restrictive / Advisory States (4 – 5)

State	Reason	Resolution
AREOR (cont..) Full Read/Write Access	• ALTER TABLE .. : • **ALTER COLUMN:** Altering the data type, length, precision, or scale of a column and DDL_MATERIALIZATION subsystem parameter is set to ALWAYS_PENDING. • **DROP COLUMN** • **ADD PARTITION -** AREOR for the newly added partition and the next logical partition • **ALTER PARTITION:** Changing the limit keys for partition-by-range (PBR) table spaces or partitioned (non-UTS) table spaces with table-controlled • **ALTER PARTITIONING (V13 FL 500):** Converting the partitioning scheme of a table (such as from a partition-by-growth to a partition-by-range scheme)	REORG SHRLEVEL CHANGE or REFERENCE **(not NONE !)**

Appendix: REORG Restrictive / Advisory States (5 – 5)

State	Reason	Resolution

AREOR (cont..)	• ALTER INDEX **.. COMPRESS**
	  • for UTS – AREOR state
	  • for non-UTS – REBUILD-pending (RBDP) status
		REORG SHRLEVEL CHANGE or REFERENCE **(not NONE !)**

Full Read/Write Access	• **Always-Pending DDL materialization:** If the Db2 subsystem parameter DDL_MATERIALIZATION is set to ALWAYS_PENDING (value 2), applicable object definition changes are never materialized immediately, and the containing table space is placed in AREOR status until a REORG is run	

Thank You