PDUG

Gdańsk, 24-25 October 2024

Clean up the table!

Why, when and how to REORG Db2 LUW tables

Kacper Kubica

HSBC Polska

POLISH DB2 USERS GROUP

Agenda

- WHY? (Reasons we need to reorg)
- WHEN? (When a REORG is needed)
- HOW? (How to reorganise a table)
- Automate your reorgs
- Best practices

Why?

Reasons we need to reorg

Reasons to reorg

- There are empty or partially **empty pages** in the table object.
- There are **overflow records**.
- We need to change the **order of records**.

Empty pages problem

- Pseudo-deletion
  - DELETE means pseudo-deletion - records are not physically deleted
  - You are left with unusable space (partially empty pages)
  - To physically delete records, **you need to reorganise the table**
- Empty pages
  - Deleting consecutive records may result in completely empty pages.
  - These pages are not automatically released, **you need to reorganise the table**

Overflow records problem

- When row size increases, records may overflow
- It means a part of the record is moved to another page, leaving a pointer on its original position
- It leads to worse I/O, which means slower read
- To remove overflow records, **you need to reorganise the table**

Order of records problem

- Inserted records are added at the end of the table
- To reorder them according to particular index, **you need to reorganise the table**

Summary

- Issues that waste disk space or degrade performance:
  - Unused (pseudo-deleted) space in data pages
  - Unused data pages
  - Overflow records
  - Unwanted order of records

PDUG

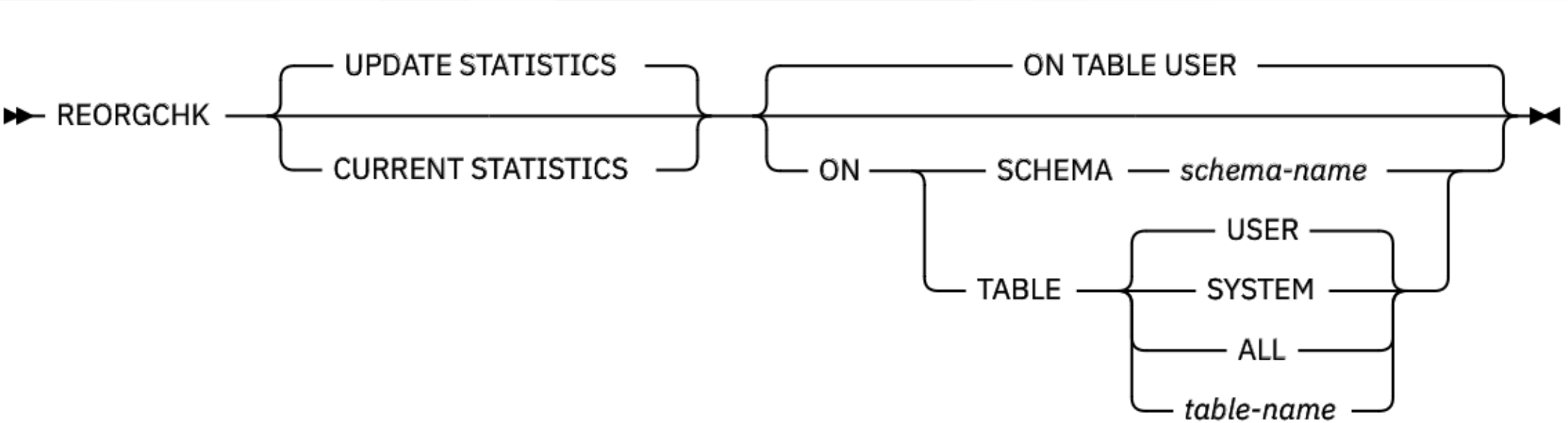
When?

How to check if a reorg is needed

Ways to check if a REORG is needed

- How to check if a table should be reorganised:
  - Create a script using:
    - SYSPROC.REORGCHK_TB_STATS procedure
    - SESSION.TB_STATS table
  - Use a tool:
    - **REORGCHK** command

REORGCHK command



REORGCHK output - table stats

```
Table statistics:

SCHEMA.NAME                CARD  OV  NP  FP ACTBLK  SIZE  F1  F2  F3 REORG
-------------------------------------------------------------------------
Table: USER1.MYDPARTT1
                              -   -   -   -      -     -   -   -   - ---

Table: USER1.MYDPARTT1
Data Partition:  PART0
                              -   -   -   -      -     -   -   -   - ---

Table: USER1.MYDPARTT1
Data Partition:  PART1
                              -   -   -   -      -     -   -   -   - ---

Table: USER1.MYDPARTT1
Data Partition:  PART2
                              -   -   -   -      -     -   -   -   - ---
-------------------------------------------------------------------------
```

» Source: IBM Db2 LUW documentation

Functions in table statistics of REORGCHK

- There are 3 functions in the in table statistics section:
  - F1 - percentage of overflow records in the table (F1 = OV / CARD)
  - F2 - percentage of allocated space used to store the data
  - F3 - percentage of pages containing some data (F3 = NP / FP)
- and REORG column that tells if:
  - F1 > 5
  - F2 < 70
  - F3 < 80

REORGCHK output - index stats

```
Index statistics:

SCHEMA.NAME                  INDCARD LEAF ELEAF LVLS NDEL KEYS ... F4 F5 F6 F7 F8 REORG
---------------------------------------------------------------- ... ------------------
Table: USER1.MYDPARTT1
Index: USER1.MYNONPARTIDX1
                                   -    -    -    -    -    - ...  -  -  -  -  -  -----

Index: USER1.MYDPARTIDX1
Data Partition:   PART0
                                   -    -    -    -    -    - ...  -  -  -  -  -  -----

Index: USER1.MYDPARTIDX1
Data Partition:   PART1
                                   -    -    -    -    -    - ...  -  -  -  -  -  -----

Index: USER1.MYDPARTIDX1
Data Partition:   PART2
                                   -    -    -    -    -    - ...  -  -  -  -  -  -----
---------------------------------------------------------------- ... ------------------
```

» Source: IBM Db2 LUW documentation

REORGCHK output before reorganisation

```
Table statistics:

F1: 100 * OVERFLOW / CARD < 5
F2: 100 * (Effective Space Utilization of Data Pages) > 70
F3: 100 * (Required Pages / Total Pages) > 80

SCHEMA.NAME                      CARD     OV     NP     FP ACTBLK    TSIZE  F1  F2  F3 REORG
-------------------------------------------------------------------------------------------
Table: PDUG_2024.TAB_F1_F2
                                58442   3329    672    719      -  1870144   5  65  94 **-
Table: PDUG_2024.TAB_F1_F3
                               100000  46304    672   1181      -  4000000  46  84  57 *-*
Table: PDUG_2024.TAB_F2_F3
                               140000      0    942   1344      -  3780000   0  70  70 -**
Table: PDUG_2024.TAB_FREE_EXTENTS
                                79000      0    531    672      -  2133000   0  79  79 --*
Table: PDUG_2024.TAB_HALF_EMPTY_PAGES
                                50000      0    672    672      -  1350000   0  50 100 -*-
Table: PDUG_2024.TAB_INDEX_PROBLEM
                               100000      0    498    498      -  2000000   0 100 100 ---
Table: PDUG_2024.TAB_OK
                               100000      0    672    672      -  2700000   0 100 100 ---
Table: PDUG_2024.TAB_OVERFLOW_ROWS
                               100000   6945    672    749      -  2900000   6  97  90 *--
-------------------------------------------------------------------------------------------
```

REORGCHK - index statistics

```
Index statistics:

F4: CLUSTERRATIO or normalized CLUSTERFACTOR > 80
F5: 100 * (Space used on leaf pages / Space available on non-empty leaf pages) > MIN(50, (100 - PCTFREE))
F6: (100 - PCTFREE) * (Amount of space available in an index with one less level / Amount of space required for all keys) < 100
F7: 100 * (Number of pseudo-deleted RIDs / Total number of RIDs) < 20
F8: 100 * (Number of pseudo-empty leaf pages / Total number of leaf pages) < 20

SCHEMA.NAME                INDCARD  LEAF ELEAF LVLS  NDEL    KEYS LEAF_RECSIZE NLEAF_RECSIZE LEAF_PAGE_OVERHEAD NLEAF_PAGE_OVERHEAD  PCT_PAGES_SAVED  F4  F5  F6  F7  F8 REORG
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
Table: PDUG_2024.TAB_INDEX_PROBLEM
Index: PDUG_2024.IDX_COL_1
                           100000   231     0    3     0    3701           5             5               666               666                0   0  92  62   0   0 *----
Index: PDUG_2024.IDX_COL_2
                           100000   225     0    3     0     774           5             5               666               666                0  14  91  64   0   0 *----
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

```
SCHEMA.NAME                INDCARD  F4  F5  F6  F7  F8 REORG
-----------------------------------------------------------
Table: PDUG_2024.TAB_INDEX_PROBLEM
Index: PDUG_2024.IDX_COL_1
                           100000   0  92  62   0   0 *----
Index: PDUG_2024.IDX_COL_2
                           100000  14  91  64   0   0 *----
-----------------------------------------------------------
```

PDUG

How?

How to reorganise a table

Classic REORG

- 2 types of REORGs: **classic** and **inplace**
- Phases of classic REORG:
  - **Sort** records (only if clustering index or reorg index specified)
  - **Build** - creating a reorganised copy
  - **Replace** - original table replaced by the copy
  - **Recreate indexes**
- Phases „build“ and „recreate indexes“ may last long
- Tables are **unmodifiable** during the classic REORG

Inplace REORG

- Allow full access to the reorganised table
- Phases of inplace REORG:
  - **Select** a range of extents
  - **Vacate** the selected range and move it to the free pages section
  - **Fill** the range with reorganised and sorted records
  - **Truncate** the table to reclaim space (only if NOTRUNCATE not specified)
- Phases „select“, „vacate“ and „fill“ run in a loop
- More flexible, but slower and less effective than classic REORG

Other REORGs

- 4 types of reorganisation:
  - classic (offline)
  - inplace:
    - full
    - **cleanup overflows**
  - **reclaim extents** (only for MDC and ITC tables)

REORGCHK output before reorganisation

```
Table statistics:

F1: 100 * OVERFLOW / CARD < 5
F2: 100 * (Effective Space Utilization of Data Pages) > 70
F3: 100 * (Required Pages / Total Pages) > 80

SCHEMA.NAME                      CARD     OV     NP     FP ACTBLK    TSIZE  F1  F2  F3 REORG
-------------------------------------------------------------------------------------------
Table: PDUG_2024.TAB_F1_F2
                                58442   3329    672    719       -  1870144   5  65  94 **-
Table: PDUG_2024.TAB_F1_F3
                               100000  46304    672   1181       -  4000000  46  84  57 *-*
Table: PDUG_2024.TAB_F2_F3
                               140000      0    942   1344       -  3780000   0  70  70 -**
Table: PDUG_2024.TAB_FREE_EXTENTS
                                79000      0    531    672       -  2133000   0  79  79 --*
Table: PDUG_2024.TAB_HALF_EMPTY_PAGES
                                50000      0    672    672       -  1350000   0  50 100 -*-
Table: PDUG_2024.TAB_INDEX_PROBLEM
                               100000      0    498    498       -  2000000   0 100 100 ---
Table: PDUG_2024.TAB_OK
                               100000      0    672    672       -  2700000   0 100 100 ---
Table: PDUG_2024.TAB_OVERFLOW_ROWS
                               100000   6945    672    749       -  2900000   6  97  90 *--
-------------------------------------------------------------------------------------------
```

Result of inplace cleanup overflows reorg

```
[db2inst1@db2server backup]$ db2 "reorg table TAB_OVERFLOW_ROWS inplace cleanup overflows"
```

```
Table statistics:

F1: 100 * OVERFLOW / CARD < 5
F2: 100 * (Effective Space Utilization of Data Pages) > 70
F3: 100 * (Required Pages / Total Pages) > 80

SCHEMA.NAME                      CARD     OV     NP     FP ACTBLK    TSIZE  F1  F2  F3 REORG
-------------------------------------------------------------------------------------------
Table: PDUG_2024.TAB_F1_F2
                                58442   3329    672    719       -  1870144   5  65  94 **-
Table: PDUG_2024.TAB_F1_F3
                               100000  46304    672   1181       -  4000000  46  84  57 *-*
Table: PDUG_2024.TAB_F2_F3
                               140000      0    942   1344       -  3780000   0  70  70 -**
Table: PDUG_2024.TAB_FREE_EXTENTS
                                79000      0    531    672       -  2133000   0  79  79 --*
Table: PDUG_2024.TAB_HALF_EMPTY_PAGES
                                50000      0    672    672       -  1350000   0  50 100 -*-
Table: PDUG_2024.TAB_INDEX_PROBLEM
                               100000      0    498    498       -  2000000   0 100 100 ---
Table: PDUG_2024.TAB_OK
                               100000      0    672    672       -  2700000   0 100 100 ---
Table: PDUG_2024.TAB_OVERFLOW_ROWS
                               100000      0    749    749       -  2800000   0  93 100 ---
-------------------------------------------------------------------------------------------
```

Result of inplace full reorg

```
[db2inst1@db2server backup]$ db2 "reorg table TAB_OVERFLOW_ROWS inplace"
```

```
Table statistics:

F1: 100 * OVERFLOW / CARD < 5
F2: 100 * (Effective Space Utilization of Data Pages) > 70
F3: 100 * (Required Pages / Total Pages) > 80

SCHEMA.NAME                      CARD     OV     NP     FP ACTBLK    TSIZE  F1  F2  F3 REORG
-------------------------------------------------------------------------------------------
Table: PDUG_2024.TAB_F1_F2
                                58442   3329    672    719       -  1870144   5  65  94 **-
Table: PDUG_2024.TAB_F1_F3
                               100000  46304    672   1181       -  4000000  46  84  57 *-*
Table: PDUG_2024.TAB_F2_F3
                               140000      0    942   1344       -  3780000   0  70  70 -**
Table: PDUG_2024.TAB_FREE_EXTENTS
                                79000      0    531    672       -  2133000   0  79  79 --*
Table: PDUG_2024.TAB_HALF_EMPTY_PAGES
                                50000      0    672    672       -  1350000   0  50 100 -*-
Table: PDUG_2024.TAB_INDEX_PROBLEM
                               100000      0    498    498       -  2000000   0 100 100 ---
Table: PDUG_2024.TAB_OK
                               100000      0    672    672       -  2700000   0 100 100 ---
Table: PDUG_2024.TAB_OVERFLOW_ROWS
                               100000      0    716    716       -  2800000   0  98 100 ---
-------------------------------------------------------------------------------------------
```

Unsuccessful try to reclaim extent

```
[db2inst1@db2server backup]$ db2 "reorg table TAB_FREE_EXTENTS reclaim extents"
SQL2221N  The REORG command failed because the specified table
"TAB_FREE_EXTENTS" is incompatible with the request to reclaim extents.
SQLSTATE=5U044
[db2inst1@db2server backup]$
```

SQL2221N The REORG command failed because the specified table *table-name* is incompatible with the request to reclaim extents.

Last Updated: 2024-02-06

Explanation

If the REORG TABLE RECLAIM EXTENTS command was issued, the specified table name is only supported when the table is a multidimensional clustered (MDC) or insert time clustering (ITC) table and the MDC or ITC table is in a database managed space (DMS) table space.

REORGCHK output before reorganisation

```
Table statistics:

F1: 100 * OVERFLOW / CARD < 5
F2: 100 * (Effective Space Utilization of Data Pages) > 70
F3: 100 * (Required Pages / Total Pages) > 80

SCHEMA.NAME                      CARD     OV     NP     FP ACTBLK    TSIZE  F1  F2  F3 REORG
-------------------------------------------------------------------------------------------
Table: PDUG_2024.TAB_F1_F2
                                58442   3329    672    719       -  1870144   5  65  94 **-
Table: PDUG_2024.TAB_F1_F3
                               100000  46304    672   1181       -  4000000  46  84  57 *-*
Table: PDUG_2024.TAB_F2_F3
                               140000      0    942   1344       -  3780000   0  70  70 -**
Table: PDUG_2024.TAB_FREE_EXTENTS
                                79000      0    531    672       -  2133000   0  79  79 --*
Table: PDUG_2024.TAB_HALF_EMPTY_PAGES
                                50000      0    672    672       -  1350000   0  50 100 -*-
Table: PDUG_2024.TAB_INDEX_PROBLEM
                               100000      0    498    498       -  2000000   0 100 100 ---
Table: PDUG_2024.TAB_OK
                               100000      0    672    672       -  2700000   0 100 100 ---
Table: PDUG_2024.TAB_OVERFLOW_ROWS
                               100000   6945    672    749       -  2900000   6  97  90 *--
-------------------------------------------------------------------------------------------
```

Comparing reorgchk after classic and inplace reorgs

```
Table statistics:

F1: 100 * OVERFLOW / CARD < 5
F2: 100 * (Effective Space Utilization of Data Pages) > 70
F3: 100 * (Required Pages / Total Pages) > 80

SCHEMA.NAME                     CARD     OV     NP     FP ACTBLK    TSIZE  F1  F2  F3 REC
------------------------------------------------------------------------------------------
Table: PDUG_2024.TAB_F1_F2
                                58442      0    473    473      -  1870144   0 100 100 --
Table: PDUG_2024.TAB_F1_F3
                               100000      0    944    944      -  3800000   0 100 100 --
Table: PDUG_2024.TAB_F2_F3
                               140000      0    941    941      -  3780000   0 100 100 --
Table: PDUG_2024.TAB_FREE_EXTENTS
                                79000      0    531    531      -  2133000   0 100 100 --
Table: PDUG_2024.TAB_HALF_EMPTY_PAGES
                                50000      0    336    336      -  1350000   0 100 100 --
Table: PDUG_2024.TAB_INDEX_PROBLEM
                               100000      0    498    498      -  2000000   0 100 100 --
Table: PDUG_2024.TAB_OK
                               100000      0    672    672      -  2700000   0 100 100 --
Table: PDUG_2024.TAB_OVERFLOW_ROWS
                               100000      0    713    713      -  2800000   0  98 100 --
------------------------------------------------------------------------------------------
```

```
Table statistics:

F1: 100 * OVERFLOW / CARD < 5
F2: 100 * (Effective Space Utilization of Data Pages) > 70
F3: 100 * (Required Pages / Total Pages) > 80

SCHEMA.NAME                     CARD     OV     NP     FP ACTBLK    TSIZE  F1  F2  F3 F
------------------------------------------------------------------------------------------
Table: PDUG_2024.TAB_F1_F2
                                58442      0    480    480      -  1870144   0  98 100 -
Table: PDUG_2024.TAB_F1_F3
                               100000      0    966    966      -  3800000   0  98 100 -
Table: PDUG_2024.TAB_F2_F3
                               140000      0    941    941      -  3780000   0 100 100 -
Table: PDUG_2024.TAB_FREE_EXTENTS
                                79000      0    531    531      -  2133000   0 100 100 -
Table: PDUG_2024.TAB_HALF_EMPTY_PAGES
                                50000      0    336    336      -  1350000   0 100 100 -
Table: PDUG_2024.TAB_INDEX_PROBLEM
                               100000      0    498    498      -  2000000   0 100 100 -
Table: PDUG_2024.TAB_OK
                               100000      0    672    672      -  2700000   0 100 100 -
Table: PDUG_2024.TAB_OVERFLOW_ROWS
                               100000      0    716    716      -  2800000   0  98 100 -
------------------------------------------------------------------------------------------
```

Reorgchk (index statistics)

```
[db2inst1@db2server backup]$ db2 reorg table tab_index_problem
DB20000I  The REORG command completed successfully.
```

```
SCHEMA.NAME                     INDCARD   F4  F5  F6  F7  F8 REORG
-------------------------------------------------------------------
Table: PDUG_2024.TAB_INDEX_PROBLEM
Index: PDUG_2024.IDX_COL_1
                                100000    0  92  62   0   0 *----
Index: PDUG_2024.IDX_COL_2
                                100000   14  91  64   0   0 *----
-------------------------------------------------------------------
```

Reorgchk (index statistics) after reorg according to index on COL_2

```
[db2inst1@db2server backup]$ db2 reorg table tab_index_problem index IDX_COL_1
DB20000I  The REORG command completed successfully.
```

```
SCHEMA.NAME                    INDCARD    F4   F5   F6   F7   F8 REORG
------------------------------------------------------------------------
Table: PDUG_2024.TAB_INDEX_PROBLEM
Index: PDUG_2024.IDX_COL_1
                               100000   100   92   62    0    0 -----
Index: PDUG_2024.IDX_COL_2
                               100000    13   91   64    0    0 *----
------------------------------------------------------------------------
```

Reorgchk (index statistics) after reorg according to index on COL_1

```
[db2inst1@db2server backup]$ db2 reorg table tab_index_problem index IDX_COL_2
DB20000I  The REORG command completed successfully.
```

```
SCHEMA.NAME                     INDCARD   F4   F5   F6   F7   F8 REORG
-------------------------------------------------------------------
Table: PDUG_2024.TAB_INDEX_PROBLEM
Index: PDUG_2024.IDX_COL_1
                                 100000    3   92   62    0    0 *----
Index: PDUG_2024.IDX_COL_2
                                 100000  100   91   64    0    0 -----
-------------------------------------------------------------------
```

Automate your reorgs

Getting REORGCHK_TB_STATS

- Calling REORGCHK_TB_STATS procedure:
  - `db2 "call SYSPROC.REORGCHK_TB_STATS('S','PDUG_2024')"`
- The procedure takes two parameters:
  - ’S’ - schema or ’T’ - table
  - Schema or table name (e.g. ’PDUG_2024’)
- Output of the procedure is saved in SESSION.TB_STATS table

Retrieving data from SESSION.TB_STATS

- Developing the SELECT expression:
  - db2 „select *** from SESSION.TB_STATS**“

Retrieving data from SESSION.TB_STATS

- Developing the SELECT expression:
  - `db2 „select * from SESSION.TB_STATS“`
  - `db2 „select substr(table_name,1,32) as TABLE, REORG from SESSION.TB_STATS“`

Retrieving data from SESSION.TB_STATS

- Developing the SELECT expression:
  - `db2 „select * from SESSION.TB_STATS“`
  - `db2 „select substr(table_name,1,32) as TABLE, REORG from SESSION.TB_STATS“`
  - `db2 „select substr(table_name,1,32) as TABLE from SESSION.TB_STATS where REORG like '%*%';“`

Collecting the list tables to reorg

- db2 „select substr(table_name,1,32) as TABLE
  from SESSION.TB_STATS where REORG like '%*%';"

```
TABLE
--------------------------------
TAB_F1_F2
TAB_F1_F3
TAB_F2_F3
TAB_FREE_EXTENTS
TAB_HALF_EMPTY_PAGES
TAB_OVERFLOW_ROWS
```

Formatting the list of tables to reorg

- Developing the CREATE TABLE expression:
  - db2 "create table TABLES_TO_REORG as (**select substr(table_name,1,32) as TABLE from SESSION.TB_STATS where REORG like** '%*%') with data";

Formatting the list of tables to reorg

- Developing the CREATE TABLE expression:
  - db2 "create table TABLES_TO_REORG as (select substr(table_name,1,32) as TABLE from SESSION.TB_STATS where REORG like '%*%') with data";
  - db2 "create table TABLES_TO_REORG as (select **ROW_NUMBER() over (order by table_name asc) as ID,** substr(table_name,1,32) as TABLE from SESSION.TB_STATS where REORG like '%*%') with data";

List of tables to reorg is ready

» db2 „select * from TABLES_TO_REORG“

```
ID                   TABLE
-------------------- ------------------------------
                   1 TAB_F1_F2
                   2 TAB_F1_F3
                   3 TAB_F2_F3
                   4 TAB_FREE_EXTENTS
                   5 TAB_HALF_EMPTY_PAGES
                   6 TAB_OVERFLOW_ROWS
```

Designing the TABLE REORG loop

- Key lines of the code related to REORGs:
  - `NUMBER_OF_TABLES_TO_REORG=$(db2 -x "select count(*) from TABLES_TO_REORG");`

Designing the TABLE REORG loop

- Key lines of the code related to REORGs:
  - `NUMBER_OF_TABLES_TO_REORG=$(db2 -x "select count(*) from TABLES_TO_REORG");`
  - **`while [ $CNT -le $NUMBER_OF_TABLES_TO_REORG ];`**

Designing the TABLE REORG loop

- Key lines of the code related to REORGs:
  - `NUMBER_OF_TABLES_TO_REORG=$(db2 -x "select count(*) from TABLES_TO_REORG");`
  - `while [ $CNT -le $NUMBER_OF_TABLES_TO_REORG ];`
    - `TABLE_NAME=$(db2 -x "select TABLE from TABLES_TO_REORG where ID=$CNT");`

Designing the TABLE REORG loop

- Key lines of the code related to REORGs:
  - `NUMBER_OF_TABLES_TO_REORG=$(db2 -x "select count(*) from TABLES_TO_REORG");`
  - `while [ $CNT -le $NUMBER_OF_TABLES_TO_REORG ];`
    - `TABLE_NAME=$(db2 -x "select TABLE from TABLES_TO_REORG where ID=$CNT");`
    - `TABLE_NAME="$(echo -e "${TABLE_NAME}" | tr -d '[:space:]')";`

Designing the TABLE REORG loop

- Key lines of the code related to REORGs:
  - `NUMBER_OF_TABLES_TO_REORG=$(db2 -x "select count(*) from TABLES_TO_REORG");`
  - `while [ $CNT -le $NUMBER_OF_TABLES_TO_REORG ];`
    - `TABLE_NAME=$(db2 -x "select TABLE from TABLES_TO_REORG where ID=$CNT");`
    - `TABLE_NAME="$(echo -e "${TABLE_NAME}" | tr -d '[:space:]')";`
    - `REORG_TABLE_STRING="reorg table PDUG_2024.${TABLE_NAME}";`

Designing the TABLE REORG loop

- Key lines of the code related to REORGs:
  - `NUMBER_OF_TABLES_TO_REORG=$(db2 -x "select count(*) from TABLES_TO_REORG");`
  - `while [ $CNT -le $NUMBER_OF_TABLES_TO_REORG ];`
    - `TABLE_NAME=$(db2 -x "select TABLE from TABLES_TO_REORG where ID=$CNT");`
    - `TABLE_NAME="$(echo -e "${TABLE_NAME}" | tr -d '[:space:]')";`
    - `REORG_TABLE_STRING="reorg table PDUG_2024.${TABLE_NAME}";`
    - **`db2 "$REORG_TABLE_STRING";`**

Designing the TABLE REORG loop

- Key lines of the code related to REORGs:
  - `NUMBER_OF_TABLES_TO_REORG=$(db2 -x "select count(*) from TABLES_TO_REORG");`
  - `while [ $CNT -le $NUMBER_OF_TABLES_TO_REORG ];`
    - `TABLE_NAME=$(db2 -x "select TABLE from TABLES_TO_REORG where ID=$CNT");`
    - `TABLE_NAME="$(echo -e "${TABLE_NAME}" | tr -d '[:space:]')";`
    - `REORG_TABLE_STRING="reorg table PDUG_2024.${TABLE_NAME}";`
    - `db2 "$REORG_TABLE_STRING";`
    - `((CNT++));`

Full ‘automated_reorg.sh’ script

```
#!/bin/bash
db2 "connect to testdb";
declare NUMBER_OF_TABLES_TO_REORG int;
declare CNT int;
declare REORG_TABLE_STRING string;
declare TABLE_NAME string;
db2 "call sysproc.reorgchk_tb_stats('S','PDUG_2024')";
db2 "create table TABLES_TO_REORG as (select ROW_NUMBER() over (order by tabl
NUMBER_OF_TABLES_TO_REORG=$(db2 -x "select count(*) from TABLES_TO_REORG");
CNT=1;
while [ $CNT -le $NUMBER_OF_TABLES_TO_REORG ];
do
TABLE_NAME=$(db2 -x "select TABLE from TABLES_TO_REORG where ID=$CNT");
TABLE_NAME="$(echo -e "${TABLE_NAME}" | tr -d '[:space:]')";
REORG_TABLE_STRING="reorg table PDUG_2024.${TABLE_NAME}";
db2 "$REORG_TABLE_STRING";
((CNT++));
done
db2 "drop table TABLES_TO_REORG";
```

REORGCHK before executing the script

PDUG

```
SCHEMA.NAME                          CARD      OV      NP      FP ACTBLK    TSIZE  F1  F2  F3 REORG
-------------------------------------------------------------------------------------------------
Table: PDUG_2024.TAB_F1_F2
                                    58442    3329     672     719      -  1870144   5  65  94 **-
Table: PDUG_2024.TAB_F1_F3
                                   100000   46304     672    1181      -  4000000  46  84  57 *-*
Table: PDUG_2024.TAB_F2_F3
                                   140000       0     942    1344      -  3780000   0  70  70 -**
Table: PDUG_2024.TAB_FREE_EXTENTS
                                    79000       0     531     672      -  2133000   0  79  79 --*
Table: PDUG_2024.TAB_HALF_EMPTY_PAGES
                                    50000       0     672     672      -  1350000   0  50 100 -*-
Table: PDUG_2024.TAB_INDEX_PROBLEM
                                   100000       0     498     498      -  2000000   0 100 100 ---
Table: PDUG_2024.TAB_OK
                                   100000       0     672     672      -  2700000   0 100 100 ---
Table: PDUG_2024.TAB_OVERFLOW_ROWS
                                   100000    6945     672     749      -  2900000   6  97  90 *--
-------------------------------------------------------------------------------------------------
```

Running the automating script

- ./automated_reorg.sh > automated_reorg.out

REORCHK after executing the script

PDUG

```
SCHEMA.NAME                          CARD      OV      NP      FP ACTBLK    TSIZE  F1  F2  F3 REORG
-------------------------------------------------------------------------------------------------
Table: PDUG_2024.TAB_F1_F2
                                    58442       0     473     473      -  1870144   0 100 100 ---
Table: PDUG_2024.TAB_F1_F3
                                   100000       0     944     944      -  3800000   0 100 100 ---
Table: PDUG_2024.TAB_F2_F3
                                   140000       0     941     941      -  3780000   0 100 100 ---
Table: PDUG_2024.TAB_FREE_EXTENTS
                                    79000       0     531     531      -  2133000   0 100 100 ---
Table: PDUG_2024.TAB_HALF_EMPTY_PAGES
                                    50000       0     336     336      -  1350000   0 100 100 ---
Table: PDUG_2024.TAB_INDEX_PROBLEM
                                   100000       0     498     498      -  2000000   0 100 100 ---
Table: PDUG_2024.TAB_OK
                                   100000       0     672     672      -  2700000   0 100 100 ---
Table: PDUG_2024.TAB_OVERFLOW_ROWS
                                   100000       0     713     713      -  2800000   0  98 100 ---
-------------------------------------------------------------------------------------------------
```

How to improve the automated reorg scripts

- The real-life automated reorg script should:
  - Be dynamic (should **NOT** have **hardcoded** database and schema names)
  - Use a method of deciding if a given table needs a reorg **tailored for your environment** (not necessarily just an ‘*’ in the REORG column)
  - Be **properly tested**
  - And probably much more…

Best practices

Best Practices

- Think about your **strategy**
- **Reorg wisely**
- **Automate** the process
- Make your **scripts dynamic**
- Constantly look for **improvements**

PDUG

Thank You!

kubicakacper@gmail.com

medium.com/@kubicakacper

linkedin.com/in/kubicakacper