Back to Basics: Db2 Logging 101 and Exploitation Beyond Recovery

BROADCOM
MAINFRAME SOFTWARE

Steen Rasmussen, Customer Services Consultant

Steen.rasmussen@Broadcom.com

Disclaimer

Certain information in this presentation may outline Broadcom's general product direction. 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 or specifications for any Broadcom software product. This presentation is based on current information and resource allocations as of April 14, 2023, 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 WARRANTIES 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.

Abstract

This is a “beginner” session describing why the LOG is one of the crucial components of Db2, and without the log there would be no way to recover corrupted objects, no way to “recover” failed SQL (delete, insert and update) among other tasks.

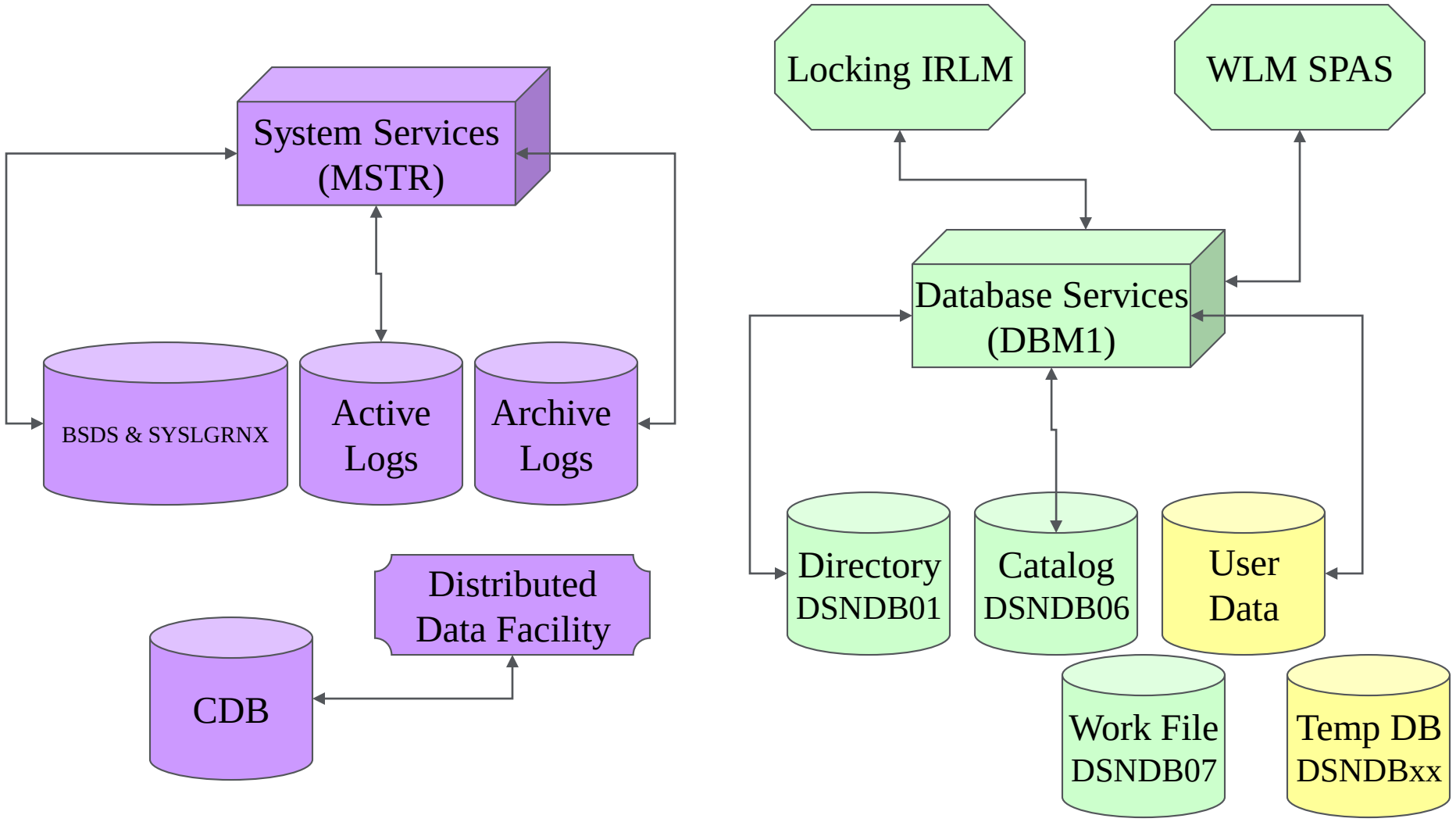
This presentation will cover some log basics and what the log holds in order to back out work - as well as what’s in and NOT in the log. Finally, once you have an understanding of what’s in the log, we will go over some use cases how you can exploit the log beyond what it is really intended for.

How can you benefit from the log-records – even without a LOG TOOL *(not my recommendation)*

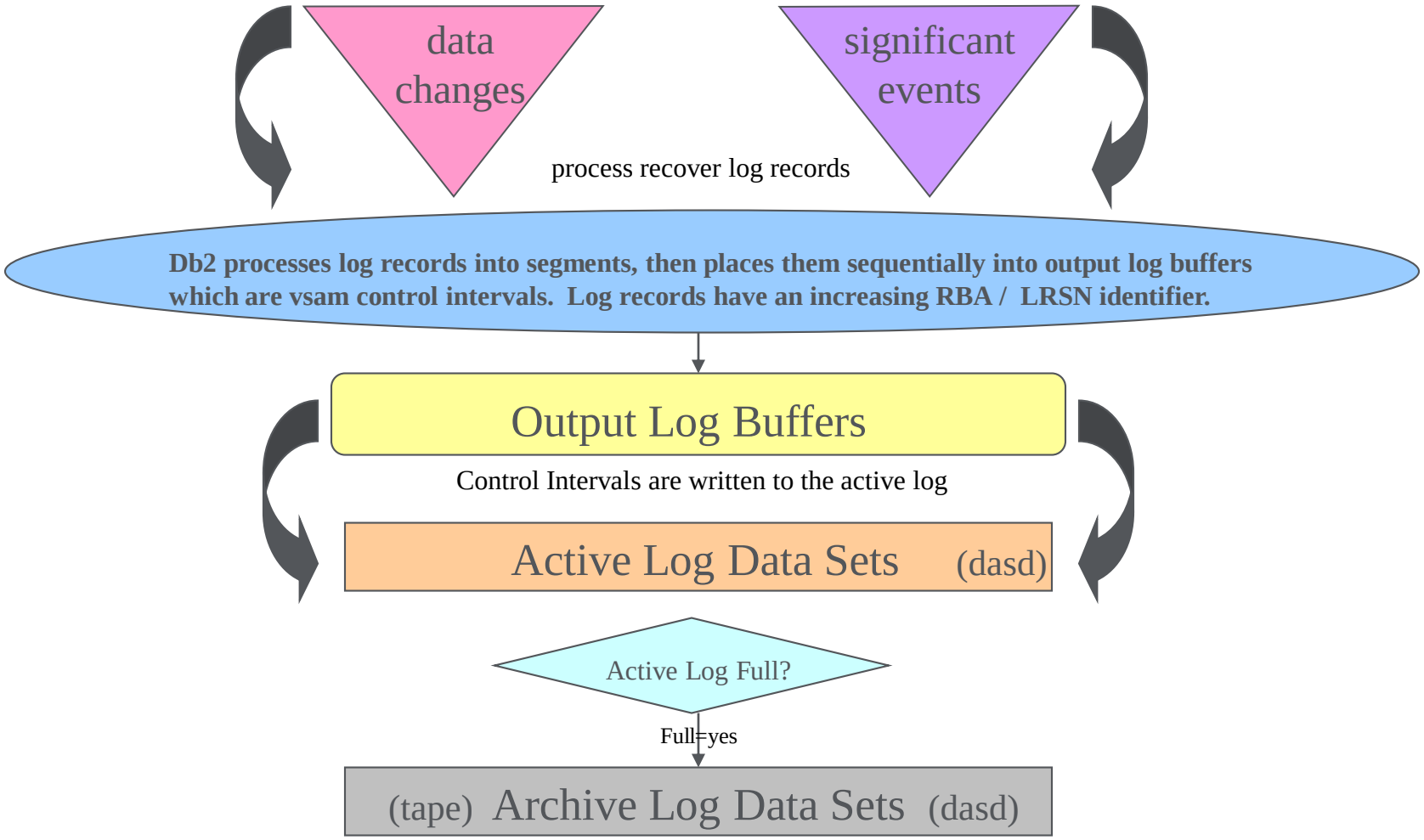
Agenda

- The Db2 Log’s role in the eco system
- What’s logged and NOT logged and the importance of DCC
- Retrieval of log information – DSN1LOGP
- Some interesting log related topics/features
- Exploitation of the log beyond recovery

Db2 Subsystem – some components



The Db2 Log – High Level Overview



If active log is full it is offloaded to a new archive log data set.

Why is Db2 Logging

- Basically two types of log-records for data changes
  - REDO : The new look
  - UNDO : The old look
- UNDO : Back out what was changed in case of an abnormal termination, some type of failure or if the executing application decides to ROLLBACK
- REDO : Recovery
  - Forward from imagecopy + logapply to a PIT or current
  - LOG-ONLY recovery if possible (for example after DSN1COPY or from when table was created).
  - Backward in case “incident” is recent this can be more efficient (will use UNDO).

What is Logged for Data Page Changes ?

- Connection Type : TSO, BATCH, UTILITY, CICS, IMS
- Connection id : established by attachment facility
  - CAF : DB2CALL, TSO : TSO or BATCH, ...
- Correlation id : associated with DB2 thread
- Authorization id, Plan name
- Date - Time
- URID, RBA, LRSN
- DBID, PSID, OBID
- Type of operation
- data..... *(and not only data itself but also header pages, compression dictionary etc.)*

What is NOT Logged for Data Page Changes ?

- Package Name
- Table Creator and Name
  - What if OBID changes due to DROP/CREATE and table name remain the same
  - You will have to scan the log to verify you don’t need TWO different OBID’s
- Index Creator and Name
- Database and Tablespace name
- Other activity NOT in the LOG:
  - SELECT statements
  - Failed access / auth failures

Db2 11 Changes to RBA/LRSN

- Optional in Db2 11
- RBA increased from 6 -> 10 bytes to avoid running out of values
- Basically a full word is added to the left of the existing RBA
- Range increases from 2(48) to 2(80)
  - From 281 Trillion Bytes - 281,474,976,701,656
  - To 1 Septillion - 1,208,925,819,614,629,174,706,176

Some topics to be aware of

CLR : Compensation Log Record (when rollback happens)

URID : Unit-Of-Recovery-Id (the activity between start of new “transaction” and COMMIT

DSNJU003 : Utility to list active/archive logs from BSDS

SYSLGRNX : Holds pageset open-close information. Used to skip reading log-ranges for recovery as an example.

MODIFY utility : Cleanup of SYSCOPY and SYSLGRNX

DSN1LOGP : Print information from the LOG – let’s dive into these details.

What table-row information gets logged ? It depends .....

- SQL qualified DELETE: The entire row
- SQL INSERT: The entire row
- SQL MASS DELETE: Spacemap page byte – unless Data Capture Changes active for the table – then each row
- SQL UPDATE: By default only partial row information logged – unless Data Capture Changes active – then complete before/after image is logged.
  - Potential concerns related to “too much logging”
  - But are all updated columns always adjacent ?
  - VARCHAR expansions etc. …….?

What table-row information gets logged ? It depends .....

Data Capture Changes impact when generating complete row image (important from an overhead perspective):

- If complete row image not present (UPDATEs)
- Two options:
  1. Read most recent full imagecopy prior to log-range needed. Then apply all log-records up to log-range of interest – then the complete WHERE clause for the UPDATE statement can be generated.
  2. Read the active VSAM pageset and scan the log-records backwards to generate the complete WHERE clause for the UPDATE.

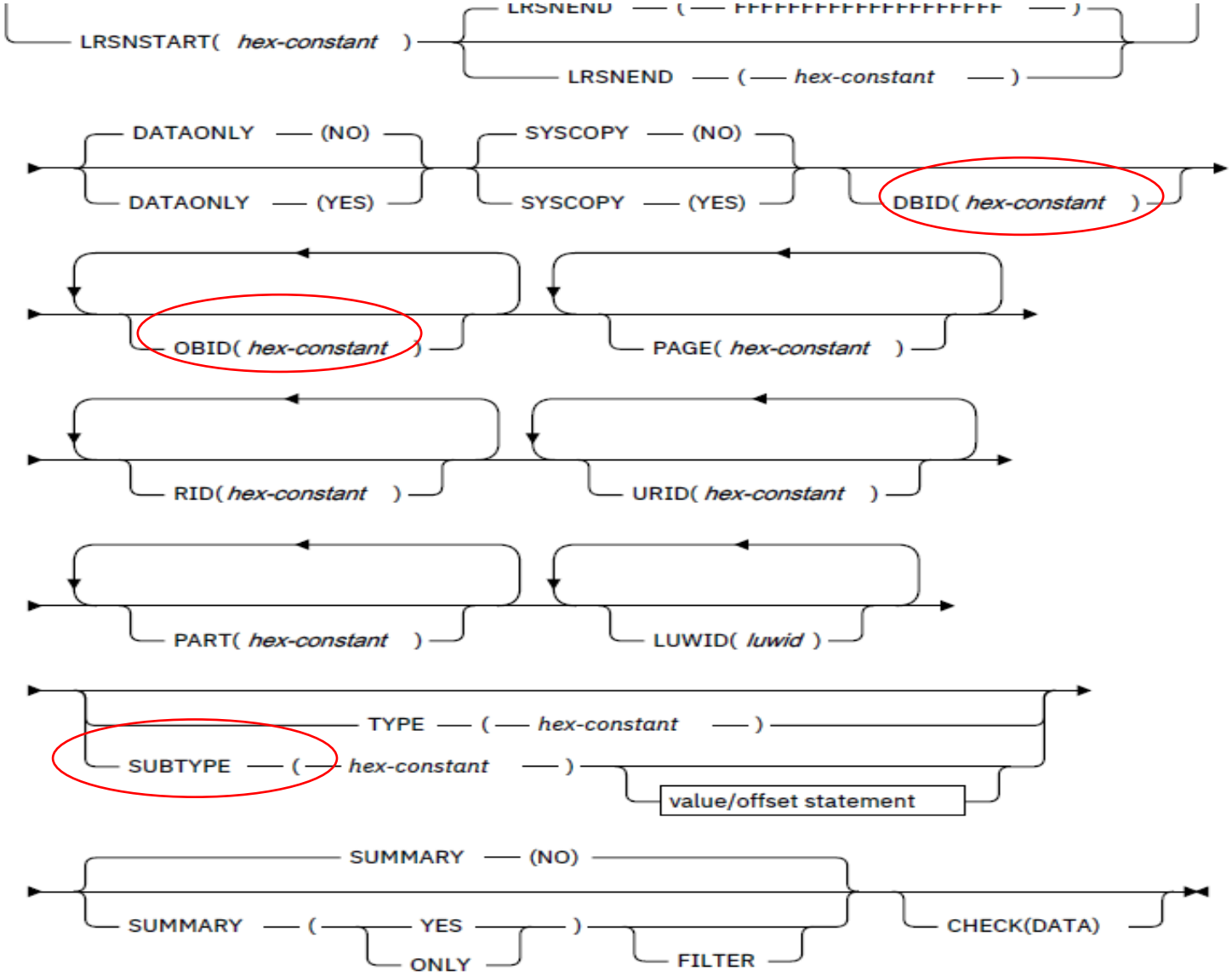
DSN1LOGP Examples

DSN1LOGP Sample JCL

The very basic JCL – you should limit by using FILTERING

```
//*************************************************************//
//* MODEL JCL FOR DSN1LOGP REPORT                             */
//*************************************************************//
//STEP1    EXEC PGM=DSN1LOGP
//*
//STEPLIB  DD DISP=SHR,DSN=D12A.PRIVATE.SDSNEXIT
//         DD DISP=SHR,DSN=DB2.DB2C10.GA.RSU2203.SDSNLOAD
//*
//SYSSUMRY DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//ABNLIGNR DD   DUMMY                  SUPPRESS ABENDAID DUMPS
//BSDS     DD DSN=D12A.BSDS01,DISP=SHR
//SYSIN    DD *
   RBASTART  (0000000019B659F7D000)
     URID    (0000000019B659F7D673)
   RBAEND    (0000000019B659F7E000)
   SUMMARY(NO)
/*
```

DSN1LOGP – Partial Syntax



DSN1LOGP Examples

- Table is created – immediately followed by two inserts.
  - Guess #log-records generated for these two insert statements ???
  - Let's verify via DSN1LOGP
    - Not all related to SQL
    - Hence filtering recommended

DSN1LOGP1.txt

```
CREATE  TABLE RASST02.IDUGLOGTB
      ( "COL01" CHAR ( 8 )
      , "COL02" INTEGER
      , "COL03" VARCHAR ( 20 )
      )
      IN STEEN.IDUGLOG ;

URID: 0000000019B659F7D673
LRSN: 00DBAA3CE09E2B0F6C00
```

```
For Table    => RASST02.IDUGLOGTB                > Row number=> 1 OF 2
Edit Mode    => C                                  Max Char   => 070
SSID: D12A ----------FETCH STATUS: COMPLETE------------------------- RASST02
OPT S N:COL01    N:COL02           N:COL03
___ I N Steen    N              61 N 12bytexxxxxx
___ I N Finn     N              59 N 08bytexx
******************************** BOTTOM OF DATA ********************************
```

DSN1LOGP Examples – Partial Logged Row

Only one byte changed – column 1 and 2 and part of column 3 “missing”

```
DO )  URID(0000000019B6693D3000)
E00)  DBID(2BD3)  OBID(0016)  PART(0001)  PAGE(00000006)            14:26:39 22.
E IN A DATA PAGE)  CLR(NO)  PROCNAME(DSNIREPR)

0 00000000 00000000 19B6693D 30000000 00000000  *   k
0 06000001 00000000 000019B6 693D30BA 000000DB  *          &
0 00000000                                      *   b' +
0 000019B6 59F7DCE1 2D408000 00010000 00000000  *h L              7
                                                *
0 00000000 001DC1A7 A7A7A7A7 A7A7A7A7 A7A7      *                  Axxxxxxxxxxx

9B6693D3000)
400)  TYPE(UR CONTROL)  SUBTYPE(BEGIN COMMIT1)                      14:26:39 22.

0 00000000 00000000 19B6693D 30000000 00000000  *   m k
0 00200002 00000000 000019B6 693D3113 000000DB  *          &
0 00000000                                      *   b JU
0 00000700 0000D9C1 E2E2E3F0 F2404040 40404040  *                RASST02
0 0000D9C1 E2E2E3F0 F240D9C3 E4E4D7F2 F0F00000  *             RASST02 RCUUP200
```

- Challenge to RE-CONSTRUCT the UPDATE WHERE clause !!

```
For Table    => RASST02.IDUGLOGTB                    > Row number=> 1 OF 2
Edit Mode    => C                                      Max Char  => 070
SSID: D12A ----------FETCH STATUS: COMPLETE------------------------- RASST02
OPT S N:COL01     N:COL02           N:COL03
___ U N Steen     N              61 N 12byteAxxxxx
___   N Finn      N              59 N 08bytexx
******************************** BOTTOM OF DATA ********************************
```

DSN1LOGP Examples – DCC Enabled

Data Capture enabled – update statement can be reconstructed from the updating URID *(no backward/forward log reading needed)*

```
)  URID(0000000019B669456633)
)  DBID(2BD3)  OBID(0019)  PART(0001)  PAGE(00000006)            14:56:22
22.167
N A DATA PAGE  -  DATA CAPTURE)  CLR(NO)


0000000 00000000 19B66945 66330000 00000000  *
6000001 00000000 000019B6 694566ED 000000DB  *          &
0000000                                      *  w
00019B6 69441A68 2B428000 00010000 00000000  *h L
                                             *
0000000 00140015 00298000 00F140D5 40F1F282  *                         1 N 12b
0000029 00000100 E2A38585 95404040 00800000  *yteBxxxxx            Steen
1A7A7A7 A7A74040 4040                        *  1 N 12byteAxxxxx
```

```
For Table    => RASST02.IDUGLOGTB2                > Row number=> 1 OF 2
Edit Mode    => C                                   Max Char  => 070
SSID: D12A ----------FETCH STATUS: COMPLETE---------------------- RASST02
OPT S N:COL01     N:COL02          N:COL03
___    N Steen     N            61 N 1 N 12byteBxxxxx
___    N Finn      N            59 N 9 N 08bytexx
******************************** BOTTOM OF DATA ********************************
```

Some Interesting But Important Topics

Triggers

- Updates from Triggering statements have special log-records.
- If you generate REDO statements from one set of tables – and need to apply these to different tables – something to validate:
  - Do the tables receiving the REDO's have the same triggers – if so DON'T generate the TRIGGERED statements (triggers fired twice).
  - If NO TRIGGERS – then remember to generate the triggered statements.

If you generate UNDO's to back out changes – consider to generate the TRIGGERED statements – but DROP the triggers prior to applying the UNDO's *(and re-create the triggers afterwards in the correct sequence).*

- You could face similar challenges for RI RULES.

NOT LOGGED Tablespaces

- Introduced several releases ago.
- Not a solution to save CPU, Elapsed-Time
- More headaches than you need.
  - COPYP when activating (otherwise point of no return)
  - SQL update failure results in RECP – if no logging there's no way for Db2 to back out
- Look for other methods like optimizing/tuning logging parameters

Log Exploitation Beyond Recovery *(Real Life Scenarios)*

Intelligent/Surgical Recovery

- Batch application being tested in production to estimate execution time – “forgot to comment-out one DELETE stmt”.

  - A few thousands rows mistakenly deleted from one table.
  - Unfortunately a parent table with many FK-tables.
  - Also “logical” relationships with IMS DB’s.
  - Incident discovered hours later
  - System wide recovery would be a disaster
  - Decision to use LOG-tool to UNDO the DELETES including FK-tables.
  - Minutes to “recover” as opposed to many hours.

- CAUTION : What if applications make decisions based on the row-existence ??

What Happened to my Row *(most common)*

- Frequent scenario where everybody scratches their head to investigate why a row the look it has
- Scan the log – not looking for all updates to the specific table but search for the column-value
- The log-record with the “bad look” has plan name and user-id.

Auditing Very Common

- Used to be audit of certain users – especially SYSADM etc.
- Certain tables holding sensitive information.
  - One issue is SELECTS are not recorded in the log
  - For SELECTS – Detector is an option.
- Lately a lot of interest in also auditing DDL changes
- Db2 Commands (like start / stop) has interest too.

“Poor Man’s” Change Propagation

- CDC and propagation tools widely used *(can be expensive in more than one way)*.
- If latency not a huge issue – and you don’t have 100’s of millions of updates per hour
- Consider using REDO log-record processing:
  - Target can be another Db2 for z/OS.
  - Even RDBMS’s on other platforms (LUW, Oracle, SQL-server,…….)

….. A recent use case

- Someone accidentally FREE'd ~10,000 Packages
- Claimed he/she was innocent
- BUT ……..

- They decided to recover catalog to another system
  - Followed by generating BIND stmts
- Remember a FREE is "just" DELETE from catalog tables
  - Could have used a LOG TOOL (or DSN1LOGP….) to "UNDO"

….. And the latest from the press

- A business application group requested a database to be drooped.
- Five weeks later they had the need to do forensic analysis
- A backup of data & DDL wasn’t performed.

- What a mess
  - Extend archive logs retention period
  - Extend imagecopy datasets retention (thank GOD they were still there)
  - Log Tool “Dropped Object Recovery” feature

How Do You Exploit These Scenarios

Scanning the LOG and reconstruction of the INSERT/UPDATE/DELETE statement and the WHERE clauses is NOT trivial.

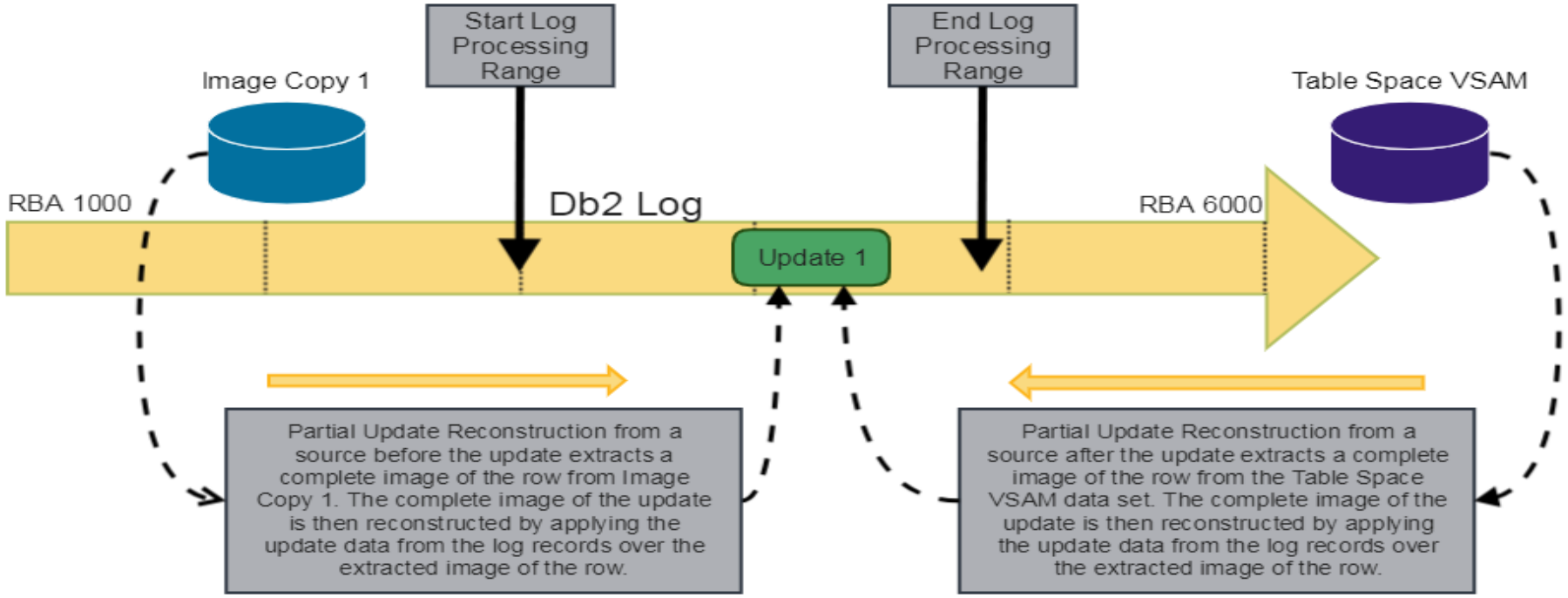
Use Broadcom's Log Analyzer, otherwise start coding based on *(highlvl.SDSNMACS(DSNDQJ00)*:

```
**   FUNCTION = MAPPING MACRO FOR THE FOLLOWING DB2 LOG RECORDS:
**
**               DSNDLRH   THE MAIN MAPPING MACRO FOR LOG RECORD
**                         HEADERS AND SEGMENTS.  ALSO CONTAINS THE
**                         CONSTANTS FOR LOG RECORD TYPES AND SUBTYPES.
**               DSNDLGOP  LOG RECORD SUBTYPE CONSTANTS (DEFINING
**                         LRHSTYPE) FOR DATA MANAGER FUNCTIONS
**               DSNDLG    DATA MANAGER LOG RECORD SUBHEADER
**               DSNDLGD   FLAG BYTE DEFINITIONS FOR LGDFL IN DSNDLG
**               DSNDLGB   DATA MANAGER LOG RECORD DESCRIBING INSERT,
**                         UPDATE, AND DELETE OF RECORDS TO A DATA PAGE
**               DSNDLGS   DATA MANAGER SEGMENTED LOG RECORD DESCRIBING
**                         OTHER CHANGES TO A DATA PAGE, E.G. LOAD AND
**                         REORG...LOG YES
**               DSNDSVPT  DATA MANAGER SAVEPOINT LOG RECORD WHICH MARKS
**                         THE BEGINNING OF A SET OF RELATED DATA CHANGE
**                         LOG RECORDS
**               DSNDLOG   PAGESET CONTROL LOG RECORDS
**               DSNDURLR  UR CONTROL LOG RECORDS
**               DSNDLUWD  LUWID DESCRIPTION.
**               DSNDLGKX  INDEX MANAGER LOG RECORDS
**               DSNDDBL   DATA MANAGER UR PAGESET CHECKPOINT LOG RECORD
**               DSNDLT    Distributed two phase commit log records.
```

Log Analyzer – IC Level No Longer Needed when DCC not Active

Partial Update Reconstruction this enhancement introduces a new mechanism to reconstruct a full image of the row when only partial data is available in the Db2 log, using the current image from the table space rather than using data from an image copy.

- No changes to the behavior of existing jobs – still recommend DML Level of Detail = I (image copy as default value)
- Data Capture Changes still most efficient even with PURSRC A (AFTER).
- Must use new parameter to benefit from the change - PURSRC which has three options: AFTER, BEFORE, and AUTO
-



BROADCOM®
MAINFRAME SOFTWARE

Thank You