SWG BetaWorks

DB2 9 for z/OS

Technical Education Series

"Reordered Row Format
What it means to you" (and how to read DB2 Logs)"

Paul Fletcher (Fletchpl@uk.ibm.com)

Important Disclaimer

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY.

WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED.

IN ADDITION, THIS INFORMATION IS BASED ON IBM’S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE.

IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION.

NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, OR SHALL HAVE THE EFFECT OF:

- CREATING ANY WARRANTY OR REPRESENTATION FROM IBM (OR ITS AFFILIATES OR ITS OR THEIR SUPPLIERS AND/OR LICENSORS); OR
- ALTERING THE TERMS AND CONDITIONS OF THE APPLICABLE LICENSE AGREEMENT GOVERNING THE USE OF IBM SOFTWARE.

2 Opinions of RRF

- **It is the way the DB2 internals work there is nothing I can do about it so what is the point looking into it.**

- **It is going to have an impact on my system so I would like to know how to tell if it is going to be a negative or positive impact**

Purpose Of presentation

- **This presentation is aimed at the people who want to be able to predict what the impact of RRF on their system is likely to be**

Agenda

- **What is Reordered Row Format**
- **Basic Row format**
  - timings of jobs
  - What is logged
- **Reordered Row format**
  - timings of jobs
  - What is logged

All Examples of logging are against Tables without DATA CAPTURE CHANGES

Reordered Row Format (RRF)

- **Automatic repositioning of Variable columns to end of row**
  - Length attributes replaced with Indicators positioned after fixed length columns
- **Any table space created in DB2 9 for z/OS NFM**
- **To Convert:**
  - REORG or LOAD REPLACE a table space or partition
  - ADD PARTITION
- **No EDITPROCs or VALIDPROCs**
- **PIT RECOVER will set the table space to the row format of the PIT**
- **Catalog / Directory remains in Basic Row Format (BRF)**

Basic Row Format

Table with Varchars in the middle.

Populated these with 5.5 million rows.

Ran Image Copy, UNLOAD and DSNTIAUL

SQL To Find Tables Still in Basic Row Format

```
SELECT DISTINCT SUBSTR(A.CREATOR,1,8) AS CREATOR,
   A.NAME, B.FORMAT
FROM SYSIBM.SYSTABLES A, SYSIBM.SYSTABLEPART B
WHERE FORMAT = '  '
AND A.DBNAME = B.DBNAME
AND A.TSNAME = B.TSNAME
AND A.CREATOR ¬= 'SYSIBM'
AND A.TYPE = 'T'
AND A.CREATOR = 'DSN8810'
```

Result of Query (Some of the Tables Still in BRF)

- **CREATOR NAME FORMAT**
- **---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------**
- **DSN8810 ACT**
- **DSN8810 AGEGROUP**
- **DSN8810 CITY**
- **DSN8810 CUSTOMER**
- **DSN8810 DEMO_UNICODE**
- **DSN8810 DEPT**
- **DSN8810 EACT**
- **DSN8810 EDEPT**
- **DSN8810 EEMP**
- **DSN8810 EEPA**
- **DSN8810 EMP**
- **DSN8810 EMPPROJACT**
- **DSN8810 EMP_PHOTO_RESUME**
- **DSN8810 EPROJ**
- **DSN8810 EPROJACT**
- **DSN8810 ETHNICGROUP**
- **DSN8810 INCOME_RANGE**
- **DSN8810 LOCATION**
- **DSN8810 MAP_TBL**
- **DSN8810 NEWDEPT**
- **DSN8810 NEWPHONE**

Columns In EEMP Table

Colum	ColNum	Datatype	Length	Scale	Null
EMPNO	**1**	**CHAR**	**6**	**0**	**N**
FIRSTNME	**2**	**VARCHAR**	**12**	**0**	**N**
MIDINIT	**3**	**CHAR**	**1**	**0**	**N**
LASTNAME	**4**	**VARCHAR**	**15**	**0**	**N**
WORKDEPT	**5**	**CHAR**	**3**	**0**	**Y**
PHONENO	**6**	**CHAR**	**4**	**0**	**Y**
HIREDATE	**7**	**DATE**	**4**	**0**	**Y**
JOB	**8**	**CHAR**	**8**	**0**	**Y**
EDLEVEL	**9**	**DECIMAL**	**5**	**0**	**Y**
SEX	**10**	**CHAR**	**1**	**0**	**Y**
BIRTHDATE	**11**	**DATE**	**4**	**0**	**Y**
SALARY	**12**	**DECIMAL**	**9**	**2**	**Y**
BONUS	**13**	**DECIMAL**	**9**	**2**	**Y**
COMM	**14**	**DECIMAL**	**9**	**2**	**Y**
RID	**15**	**CHAR**	**4**	**0**	**Y**
TSTAMP	**16**	**TIMESTMP**	**10**	**0**	**Y**

Print of Basic Row Format Tablespace

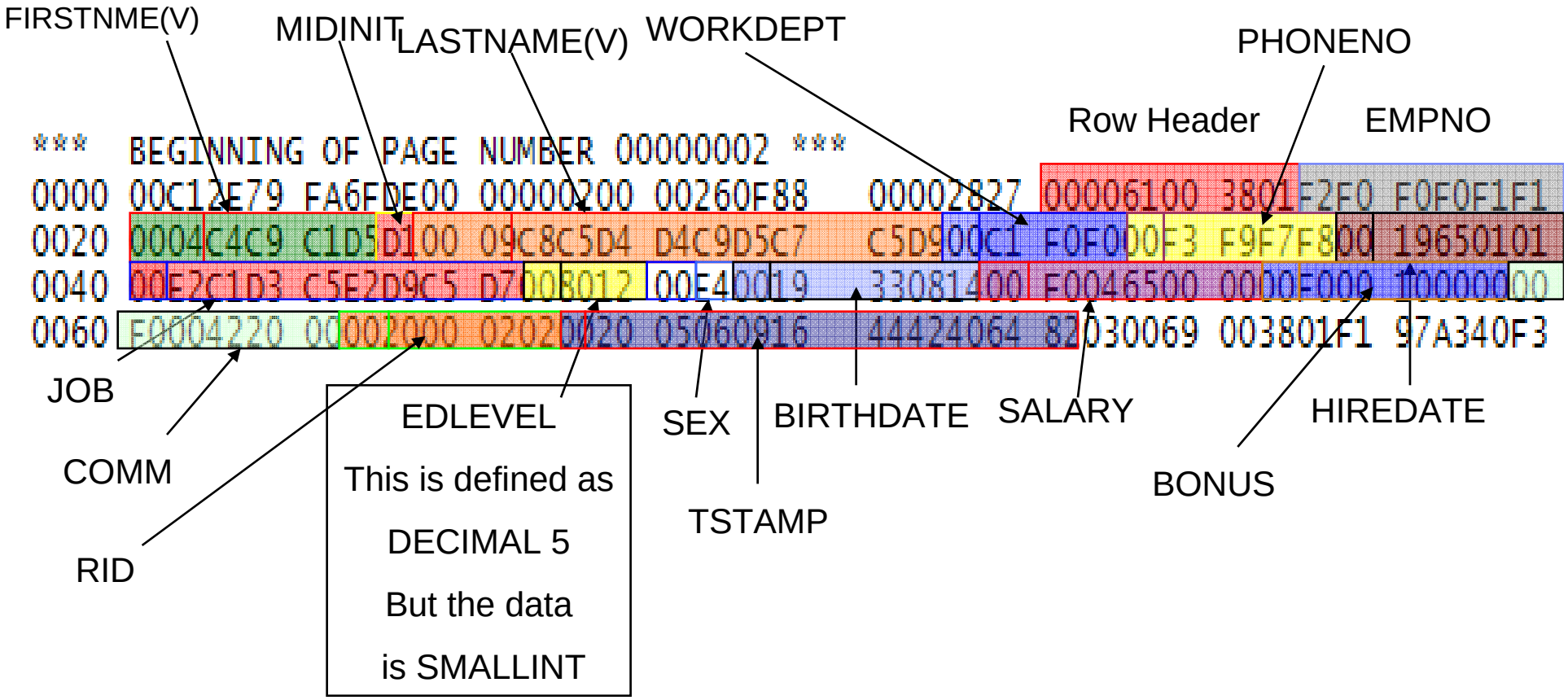


Image Copy Timings

STEPNAME	PROCSTEP	RC	EXCP	CPU	SRB	CLOCK	SERV
COPY1	DSNUPROC	00	20399	.03	.00	.90	5805K

TOTAL CPU TIME= .03 TOTAL ELAPSED TIME= .90

NUMBER OF PAGES=138691

AVERAGE PERCENT FREE SPACE PER PAGE = 0.74

PERCENT OF CHANGED PAGES = 0.01

ELAPSED TIME=00:00:50

UNLOAD Utility Timings

```
PROCSTEP    RC    EXCP     CPU     SRB  CLOCK   SERV
DSNUPROC    00   24172     .47     .00   1.04 41030K

 TOTAL CPU TIME=    .47  TOTAL ELAPSED TIME=  1.04

 UNLOAD TABLESPACE DSN8D81A.DSN8S81R
UNLOAD PHASE STATISTICS - NUMBER OF RECORDS UNLOADED=0 FOR TABLE DSN8810.EDEPT
UNLOAD PHASE STATISTICS - NUMBER OF RECORDS UNLOADED=5500001 FOR TABLE

UNLOAD PHASE STATISTICS - NUMBER OF RECORDS UNLOADED=0 FOR TABLE DSN8810.EPROJ
UNLOAD PHASE STATISTICS - NUMBER OF RECORDS UNLOADED=0 FOR TABLE DSN8810.EACT
UNLOAD PHASE STATISTICS - NUMBER OF RECORDS UNLOADED=0 FOR TABLE

UNLOAD PHASE STATISTICS - NUMBER OF RECORDS UNLOADED=0 FOR TABLE DSN8810.EEPA
UNLOAD PHASE STATISTICS - NUMBER OF RECORDS UNLOADED=5500001 FOR TABLESPACE

UNLOAD PHASE COMPLETE, ELAPSED TIME=00:01:00
```

DSNTIAUL Timings

```
 PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV  PG   PAGE   SWAP    VIO
 UNLD0       00  22814    .80    .00   1.12 59956K   0      0      0      0
NAME-FLETCHP             TOTAL CPU TIME=   .80  TOTAL ELAPSED TIME=  1.12

DSNT490I SAMPLE DATA UNLOAD PROGRAM
DSNT503I UNLOAD DATA SET SYSPUNCH RECORD LENGTH SET TO    80
DSNT504I UNLOAD DATA SET SYSPUNCH BLOCK SIZE SET TO 27920
DSNT503I UNLOAD DATA SET SYSREC00 RECORD LENGTH SET TO   134
DSNT504I UNLOAD DATA SET SYSREC00 BLOCK SIZE SET TO 32696
DSNT495I SUCCESSFUL UNLOAD      5500001 ROWS OF TABLE "DSN8810"."EEMP"
```

Update of Decimal Column

Colum	ColNum	Datatype	Length	Scale	Null
EMPNO	1	CHAR	6	0	N
FIRSTNME	2	VARCHAR	12	0	N
MIDINIT	3	CHAR	1	0	N
LASTNAME	4	VARCHAR	15	0	N
WORKDEPT	5	CHAR	3	0	Y
PHONENO	6	CHAR	4	0	Y
HIREDATE	7	DATE	4	0	Y
JOB	8	CHAR	8	0	Y
EDLEVEL	9	DECIMAL	5	0	Y
SEX	10	CHAR	1	0	Y
BIRTHDATE	11	DATE	4	0	Y
SALARY	12	DECIMAL	9	2	Y
BONUS	13	DECIMAL	9	2	Y
COMM	14	DECIMAL	9	2	Y
RID	15	CHAR	4	0	Y
TSTAMP	16	TIMESTMP	10	0	Y

Update Statement

```
UPDATE DSN8810.EEMP
SET SALARY = SALARY + 1000
WHERE SALARY = 95652.58
```

What Does DB2 Log For this Update – DSN1LOGP Output

```
0000DAA1B03D  TYPE( UNDO  REDO )  URID(0000DAA1AF0A)
              LRSN(C139E7FB8096)  DBID(0104)  OBID(0006)  PAGE(00000002)      10:5
              SUBTYPE(UPDATE IN-PLACE IN A DATA PAGE)  CLR(NO)
              PROCNAME(DSNIREPR)

   *LRH* 0048003D 06000001 0E800000 DAA1AF0A 0000DAA1 B0000626 0000DAA1 B000C139
         E7FB8096 0001
   *LG** 80010400 06000000 0200C12E 79FA6FDE 2D00
   0000  00100102 00388200 004A6652 58565258
```

What do all these Hex Characters mean???

Log Print

- **The Log Print Record is split into 4 parts**
  - Summary Information
  - Log Record Header
  - Log Record Sub Header
  - Logged Data

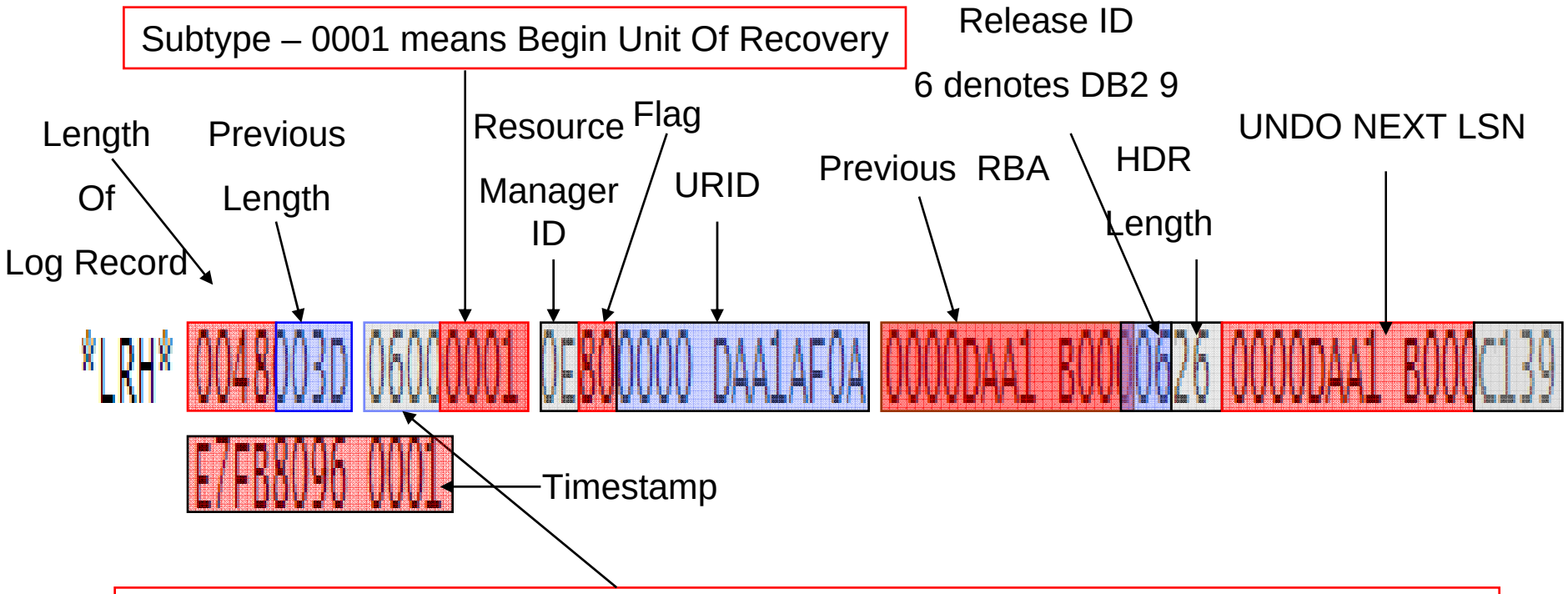
Mapping can be found in member DSNDQJ00 in the SDSNMACS Library

Summary Information

```
TYPE( UNDO  REDO )  URID(0000DAA1AF0A)
LRSN(C139E7FB8096)  DBID(0104)  OBID(0006)  PAGE(00000002)
SUBTYPE(UPDATE IN-PLACE IN A DATA PAGE)  CLR(NO)
PROCNAME(DSNIREPR)
```

This information is a summary of what is held in some of the other parts of the log record

Log Record Header



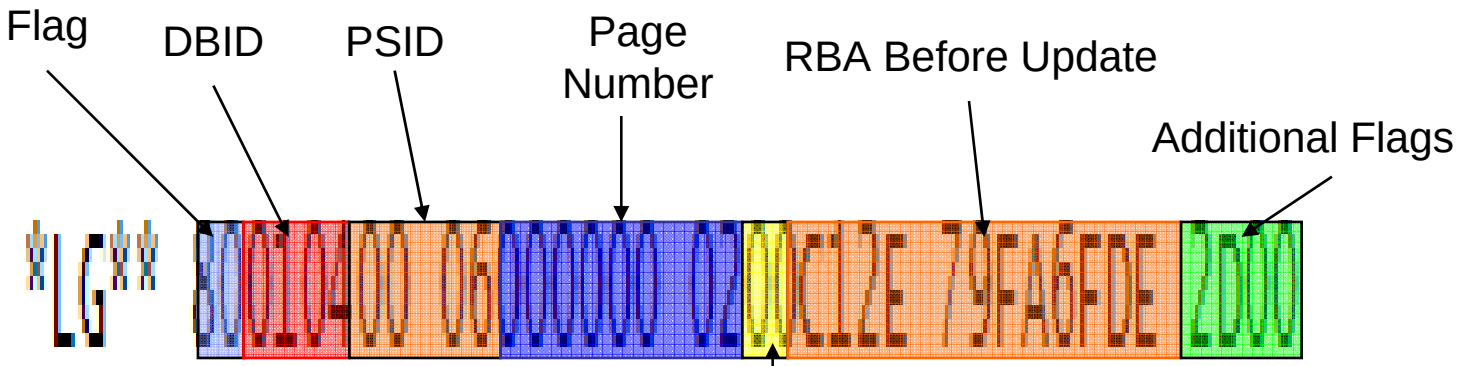
Record Type is similar to a bitmap this field has two meanings

0200 – Unit of recovery – UNDO

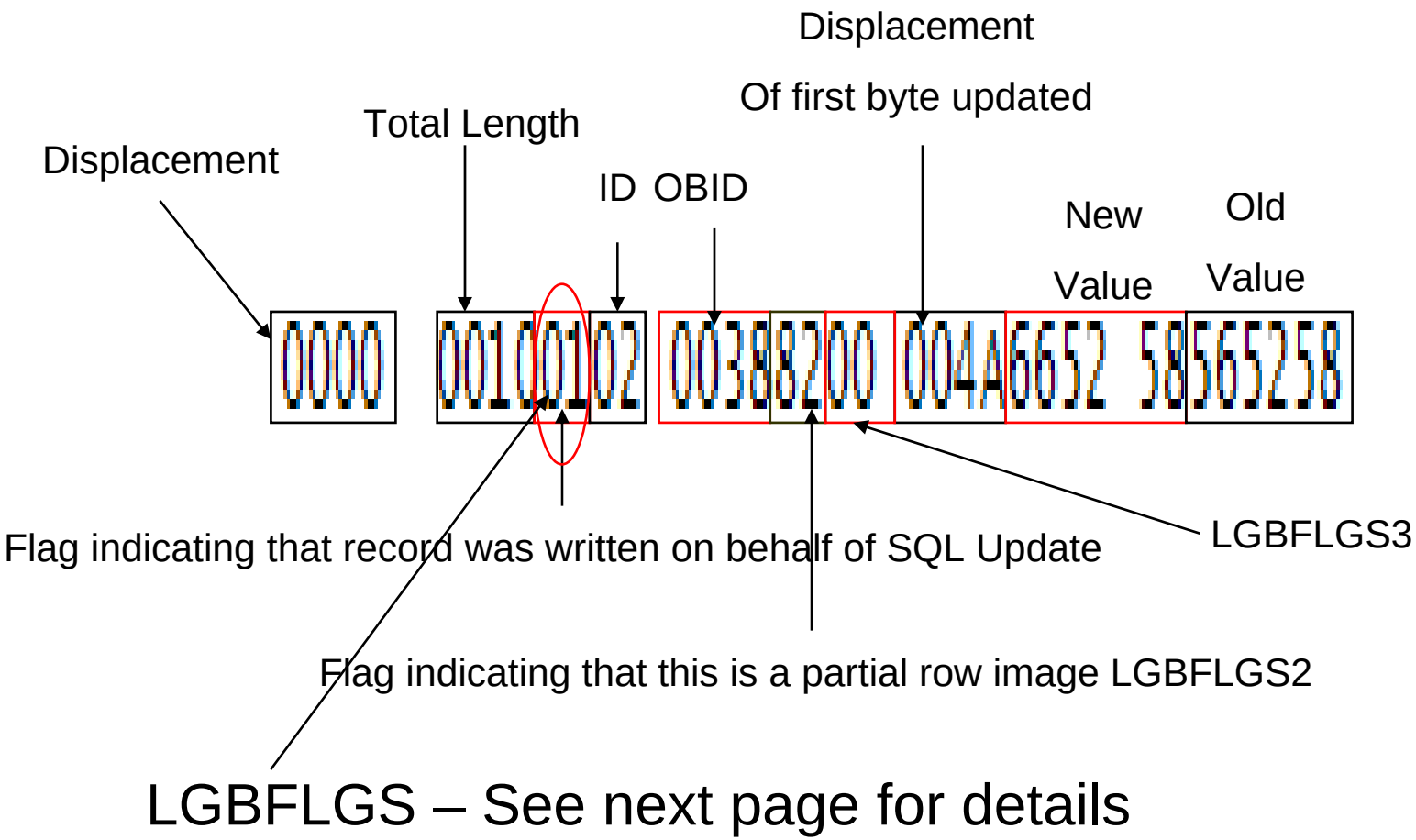
0400 – Unit of recovery – REDO

So this log record is an UNDO/REDO record

Log Record Sub Header



Log Record Data



LGBFLGS- Meaning of Bitmap

B'10000000'	YES IF UPDATE TO HASH ANCHOR NO IF RECORD CHANGE.
B‘zXXzzzzz'	TYPE OF SUB-OPERATION: (LGBSUBOP) XX = 00 IF IN-PLACE RECORD UPDATE OR HASH-ANCHOR UPDATE. 10 IF INSERT 01 IF DELETE 11 IF NON-IN PLACE REC UPDATE
B'01000000'	YES IF RECORD INSERTION
B'00100000'	YES IF RECORD DELETION
B'00010000'	YES IF ID-MAP ENTRY ADDED OR DELETED FOR RECORD ISRT/DLET
B'00001100'	TYPE OF DATA CAPTURE OPERATIONS
B'00001000'	BIT =1, TABLE IS DEFINED FOR DATA CAPTURE.
B'00000100'	BIT =1, RECURSIVE DATA CAPTURE OPERATION NOT DONE.
B'00000010'	BIT =1, RI CAUSED REPLACE OR DELETE.
B'00000001'	YES IF LOG RECORD IS WRITTEN ON BEHALF OF AN SQL UPDATE OPERATION. YES AND LGBSUBOP = '10' - LOG RECORD REPRESENTS THE INSERTION OF AN OVERFLOW RECORD ON BEHALF OF AN SQL UPDATE OPERATION.

LGBFLGS2- Meaning of Bitmap

B'10000000'	1 IF PARTIAL IMAGE LOGGED DOES NOT INCLUDE PREFIX DIFFERENCES IN PGSLTH (DATA RECORD LENGTH).
B'01000000'	1 IF RECORD HAS COMPRESSED DATA.
B'00100000'	1 IF RECORD HAS EXPANDED FORMAT LIKE CDC LOG RECORDS.
B'00010000'	1 IF GROSS LOCK IS ON
B'00001111'	VARIATION NUMBER:

- 0000 INSERT OR DELETE LOG RECORD
- 0001 UPDATE VARIATION 1 LOG RECORD
- 0010 UPDATE VARIATION 2 LOG RECORD
- 0011 UPDATE VARIATION 3 LOG RECORD
- 0100 UPDATE VARIATION 4 LOG RECORD (EXCEPT 1ST LOG REC IN SERIES)
- 0101 UPDATE VARIATION 5 LOG RECORD (EXCEPT 1ST LOG REC IN SERIES)
- 0110 UPDATE VARIATION 6 LOG RECORD (EXCEPT 1ST LOG REC IN SERIES)
- 0111 UPDATE VARIATION 7 LOG RECORD (EXCEPT 1ST LOG REC IN SERIES)
- 1000 UPDATE VARIATION 8 LOG RECORD (EXCEPT 1ST LOG REC IN SERIES)
- 1001 UPDATE VARIATION 9 LOG REOCRD
- 1110 1ST LOG RECORD IN A VARIATION 4 OR 5 SERIES
- 1111 1ST LOG RECORD IN A VARIATION 6, 7 OR 8 SERIES

What Kind of Log Record ?

- **UNDO/REDO**
- **In Place Update**
- **Partial Image**
- **Variation 2 Update Record**

In Place Updates have just one length field representing both before and after images

So what do we now know after reading the log

- **The data was on page 2**
- **The second ID on the page points to the Row**
- **The first changed byte is x'4A' bytes from the start of the row**
- **The log record held redo and undo data as a partial image**
- **Old data was Hex'565258'**
- **New data is Hex'665258'**
- **But we don't know what the rest of the column contains - yet**

Finding the column in a print taken before the update

First find Page 2

```
***  BEGINNING OF PAGE NUMBER 00000002 ***
```

Then find ID 2 by looking at the last line for the page

```
0FC0 0BAB0B49 0AE50A85 0A2509BB 095D0902   08A00840 07D50774 071206A9 064205D9
0FE0 0578051A 04BE045A 03F80395 032D02CB   026A0209 01A80143 00DE0075 0014D0C5
```

ID1

ID2

Finding the displacement of the first changed byte

- **ID2 points to displacement Hex'0075' from the start of the page**
- **So to find the displacement of the first changed byte we must add Hex'004A' to Hex'0075'**
- **This now tells us that the first changed byte is at displacement Hex'BF'**
- **So let's look at the DSN1PRNT of the tablespace and locate Hex'BF'**

Locating the column before the change

```
***  BEGINNING OF PAGE NUMBER 00000002 ***
0000 00C12E79 FA6FDE00 00000200 00260F88   00002827 00006100 3801F2F0 F0F0F1F1
0020 0004C4C9 C1D5D100 09C8C5D4 D4C9D5C7   C5D900C1 F0F000F3 F9F7F800 19650101
0040 00E2C1D3 C5E2D9C5 D7008012 00E40019   33081400 F0046500 0000F000 10000000
0060 F0004220 00002000 02020020 05060916   44424064 82030069 003801F1 97A340F3
0080 D10007A4 F994E6C2 C840D100 0DF7F885   91E6F1D3 92C29881 A48500C4 A4400098
00A0 84C8C500 20040606 0083D8E6 A78696F6   C300F773 98008400 20010702 00F00956
00C0 525800F0 14989264 000FBDD9 DFAF0093   F583A900 19980603 13221800 00000300
00E0 65003801 D4D74BD8 E3C1000C 89F4C9F1   97989440 40A3F783 F1000485 D760C400
0100 C4D59700 9884C8C5 00200406 060083D8   E6A78696 F6C300F7 73980084 00200107
```

Rest of change

First Changed byte at BF

Where is the rest of the column ?

- **Column - Salary is defined as DECIMAL(9,2)**
- **This is held in the data page as 5 bytes**
- **The full field contained Hex'F009565258'**
- **But DB2 only logged from the first changed byte to the end of the column**

Update of Varchar Column

Colum	ColNum	Datatype	Length	Scale	Null
EMPNO	1	CHAR	6	0	N
FIRSTNME	2	VARCHAR	12	0	N
MIDINIT	3	CHAR	1	0	N
LASTNAME	4	VARCHAR	15	0	N
WORKDEPT	5	CHAR	3	0	Y
PHONENO	6	CHAR	4	0	Y
HIREDATE	7	DATE	4	0	Y
JOB	8	CHAR	8	0	Y
EDLEVEL	9	DECIMAL	5	0	Y
SEX	10	CHAR	1	0	Y
BIRTHDATE	11	DATE	4	0	Y
SALARY	12	DECIMAL	9	2	Y
BONUS	13	DECIMAL	9	2	Y
COMM	14	DECIMAL	9	2	Y
RID	15	CHAR	4	0	Y
TSTAMP	16	TIMESTMP	10	0	Y

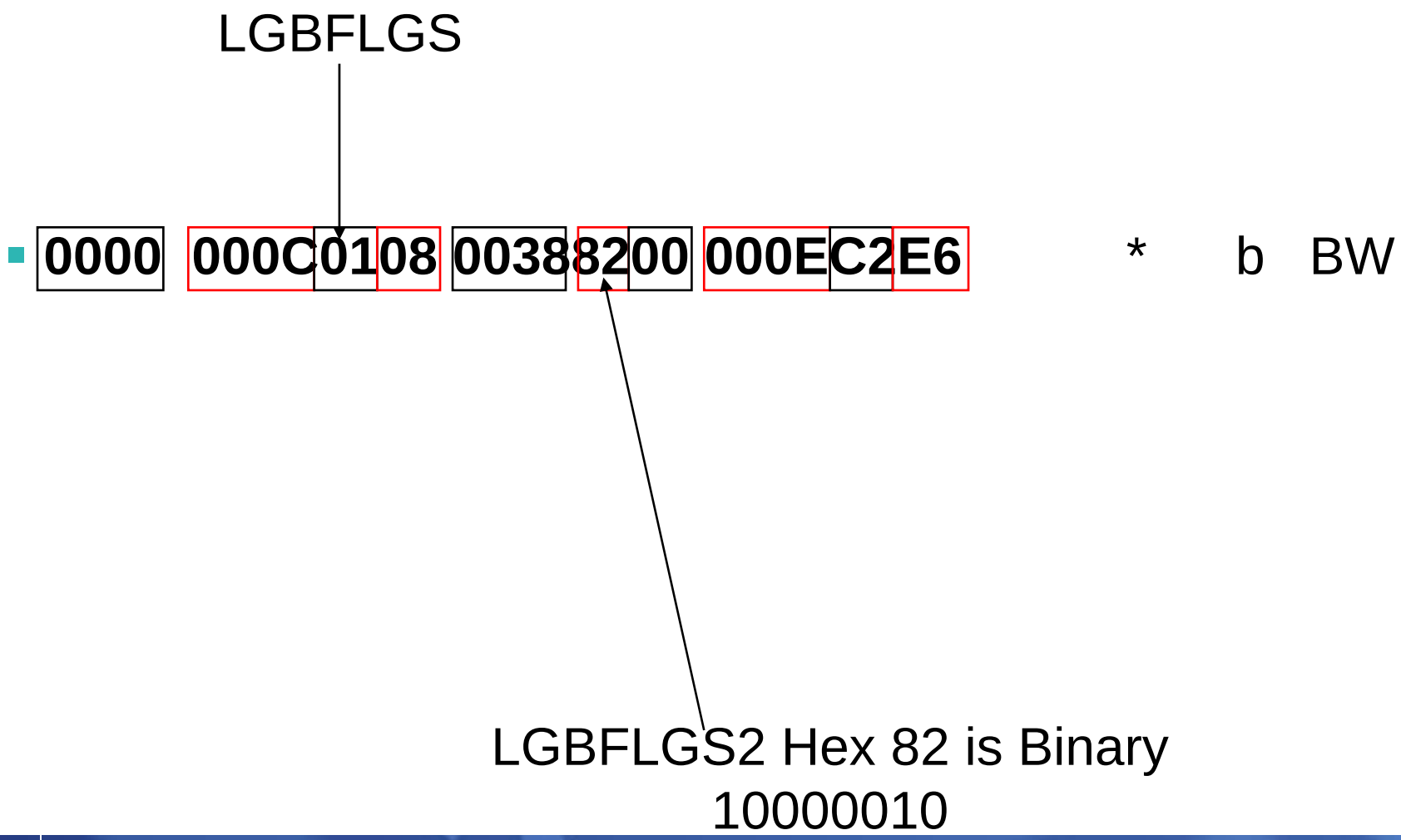
First Update – Leave the length the same

```
UPDATE DSN8810.EEMP
SET FIRSTNME = 'B'
WHERE FIRSTNME = 'W'
```

Log Header

- **0000E4B9A1CE TYPE(UNDO REDO) URID(0000E4B9A0D2)**
- **LRSN(C14CF7960786) DBID(0104) OBID(0006) PAGE(00000002)**
- **SUBTYPE(UPDATE IN-PLACE IN A DATA PAGE) CLR(NO)**
- **PROCNAME(DSNIREPR)**

What are the flags set to



LGBFLGS- Meaning of Bitmap

- **B'10000000'** **YES IF UPDATE TO HASH ANCHOR**
- **NO IF RECORD CHANGE.**
- **B‘zXXzzzzz'** **TYPE OF SUB-OPERATION: (LGBSUBOP) XX =**
- **00 IF IN-PLACE RECORD UPDATE OR HASH-ANCHOR UPDATE.**
- **10 IF INSERT**
- **01 IF DELETE**
- **11 IF NON-IN PLACE REC UPDATE**
- **B'01000000'** **YES IF RECORD INSERTION**
- **B'00100000'** **YES IF RECORD DELETION**
- **B'00010000'** **YES IF ID-MAP ENTRY ADDED OR DELETED FOR RECORD ISRT/DLET**
- **B'00001100'** **TYPE OF DATA CAPTURE OPERATIONS**
- **B'00001000'** **BIT =1, TABLE IS DEFINED FOR DATA CAPTURE.**
- **B'00000100'** **BIT =1, RECURSIVE DATA CAPTURE OPERATION NOT DONE.**
- **B'00000010'** **BIT =1, RI CAUSED REPLACE OR DELETE.**
- **B'00000001'** **YES IF LOG RECORD IS WRITTEN ON BEHALF OF AN SQL UPDATE OPERATION.**
- **YES AND LGBSUBOP = '10' - LOG RECORD REPRESENTS THE INSERTION OF AN OVERFLOW RECORD ON BEHALF OF AN SQL UPDATE OPERATION.**

LGBFLGS2- Meaning of Bitmap

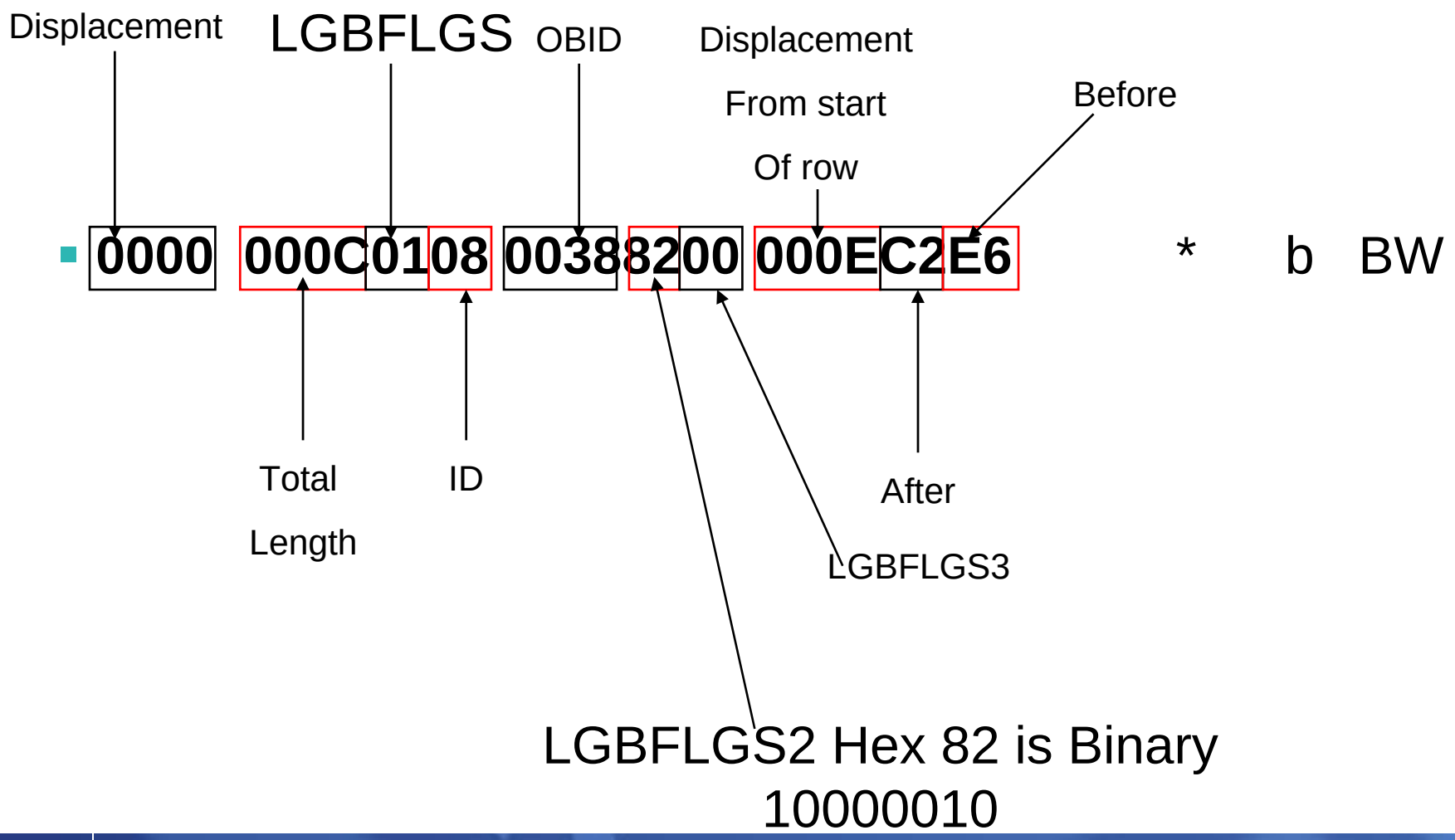
- **B'10000000' 1 IF PARTIAL IMAGE LOGGED DOES NOT INCLUDE PREFIX DIFFERENCES IN PGSLTH (DATA RECORD LENGTH).**
- **B'01000000' 1 IF RECORD HAS COMPRESSED DATA.**
- **B'00100000' 1 IF RECORD HAS EXPANDED FORMAT LIKE CDC LOG RECORDS.**
- **B'00010000' 1 IF GROSS LOCK IS ON**
- **B'00001111' VARIATION NUMBER:**
- **0000 INSERT OR DELETE LOG RECORD**
- **0001 UPDATE VARIATION 1 LOG RECORD**
- **0010 UPDATE VARIATION 2 LOG RECORD**
- **0011 UPDATE VARIATION 3 LOG RECORD**
- **0100 UPDATE VARIATION 4 LOG RECORD (EXCEPT 1ST LOG REC IN SERIES)**
- **0101 UPDATE VARIATION 5 LOG RECORD (EXCEPT 1ST LOG REC IN SERIES)**
- **0110 UPDATE VARIATION 6 LOG RECORD (EXCEPT 1ST LOG REC IN SERIES)**
- **0111 UPDATE VARIATION 7 LOG RECORD (EXCEPT 1ST LOG REC IN SERIES)**
- **1000 UPDATE VARIATION 8 LOG RECORD (EXCEPT 1ST LOG REC IN SERIES)**
- **1001 UPDATE VARIATION 9 LOG REOCRD**
- **1110 1ST LOG RECORD IN A VARIATION 4 OR 5 SERIES**
- **1111 1ST LOG RECORD IN A VARIATION 6, 7 OR 8 SERIES**

What Kind of Log Record ?

- **UNDO/REDO**
- **In Place Update**
- **Partial Image**
- **Variation 2 Update Record**

As the length stays the same the length is not written to the log

What Is Logged for Update keeping length same



Second Update Increase the length

```
UPDATE DSN8810.EEMP
SET FIRSTNME = 'HARRY'
WHERE FIRSTNME = 'H'
```

What is logged after increase in length (Summary)

0000E6B1CCAF TYPE(UNDO REDO) URID(0000E6B1CBF0)

LRSN(C14D089815F3) DBID(0104) OBID(0006) PAGE(00000002)

SUBTYPE(UPDATE NOT IN-PLACE , DATA PART ONLY IN A DATA PAGE)

CLR(NO) PROCNAME(DSNIREPR)

Log Record - Data Record

LGBFLGS2 Hex 82 is Binary 10000010

```
0000  00BA610F 00388200 000D0058 005405C8 C1D9D9E8 E9000BD6 A9E8E6F7 89D5E64B
0020  D9F10086 96D20098 84C8C500 20040606 0083D8E6 A78696F6 C300F773 98008400
0040  20010702 00F05466 614600F0 76449130 000FBDD9 DFAF0093 F583A900 19980603
0060  13221800 000001C8 E9000BD6 A9E8E6F7 89D5E64B D9F10086 96D20098 84C8C500
0080  20040606 0083D8E6 A78696F6 C300F773 98008400 20010702 00F05466 614600F0
00A0  76449130 000FBDD9 DFAF0093 F583A900 19980603 13221800 0000
```

LGBFLGS Hex 61 is Binary 01100001

LGBFLGS- Meaning of Bitmap

B'10000000' YES IF UPDATE TO HASH ANCHOR
NO IF RECORD CHANGE.

B‘zXXzzzzz' TYPE OF SUB-OPERATION: (LGBSUBOP) XX =
00 IF IN-PLACE RECORD UPDATE OR HASH-ANCHOR UPDATE.
10 IF INSERT
01 IF DELETE
11 IF NON-IN PLACE REC UPDATE

B'01000000' YES IF RECORD INSERTION

B'00100000' YES IF RECORD DELETION

B'00010000' YES IF ID-MAP ENTRY ADDED OR DELETED FOR RECORD ISRT/DLET

B'00001100' TYPE OF DATA CAPTURE OPERATIONS

B'00001000' BIT =1, TABLE IS DEFINED FOR DATA CAPTURE.

B'00000100' BIT =1, RECURSIVE DATA CAPTURE OPERATION NOT DONE.

B'00000010' BIT =1, RI CAUSED REPLACE OR DELETE.

B'00000001' YES IF LOG RECORD IS WRITTEN ON BEHALF OF AN SQL UPDATE OPERATION.

YES AND LGBSUBOP = '10' - LOG RECORD REPRESENTS THE INSERTION OF AN OVERFLOW RECORD ON BEHALF OF AN SQL UPDATE OPERATION.

LGBFLGS2- Meaning of Bitmap

B'10000000'	1 IF PARTIAL IMAGE LOGGED DOES NOT INCLUDE PREFIX DIFFERENCES IN PGSLTH (DATA RECORD LENGTH).
B'01000000'	1 IF RECORD HAS COMPRESSED DATA.
B'00100000'	1 IF RECORD HAS EXPANDED FORMAT LIKE CDC LOG RECORDS.
B'00010000'	1 IF GROSS LOCK IS ON
B'00001111'	VARIATION NUMBER:

0000 INSERT OR DELETE LOG RECORD

0001 UPDATE VARIATION 1 LOG RECORD

0010 UPDATE VARIATION 2 LOG RECORD

0011 UPDATE VARIATION 3 LOG RECORD

0100 UPDATE VARIATION 4 LOG RECORD (EXCEPT 1ST LOG REC IN SERIES)

0101 UPDATE VARIATION 5 LOG RECORD (EXCEPT 1ST LOG REC IN SERIES)

0110 UPDATE VARIATION 6 LOG RECORD (EXCEPT 1ST LOG REC IN SERIES)

0111 UPDATE VARIATION 7 LOG RECORD (EXCEPT 1ST LOG REC IN SERIES)

1000 UPDATE VARIATION 8 LOG RECORD (EXCEPT 1ST LOG REC IN SERIES)

1001 UPDATE VARIATION 9 LOG REOCRD

1110 1ST LOG RECORD IN A VARIATION 4 OR 5 SERIES

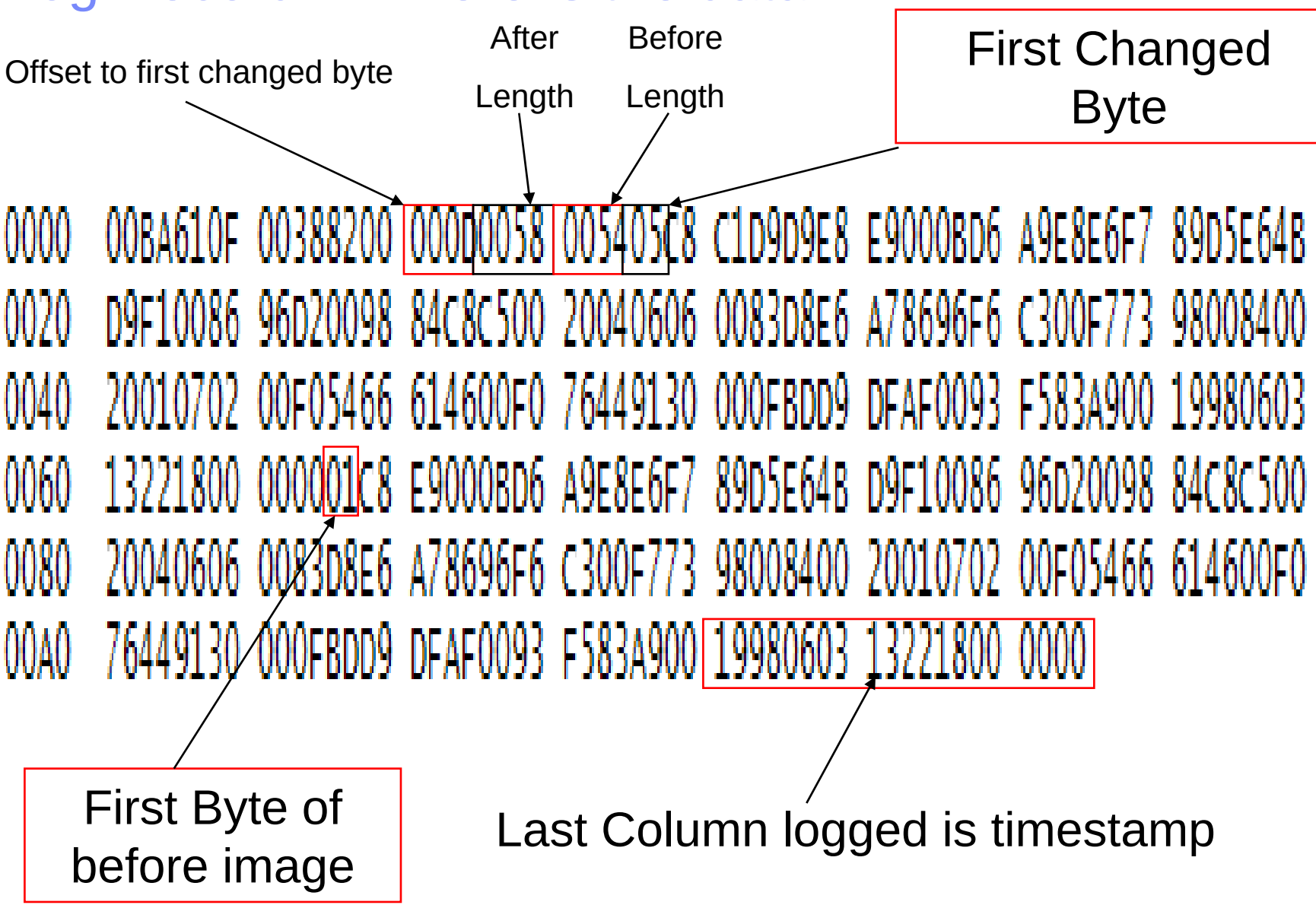
1111 1ST LOG RECORD IN A VARIATION 6, 7 OR 8 SERIES

What Kind of Log Record ?

- **UNDO/REDO**
- **Non In Place Update**
- **Partial Image**
- **Variation 2 Update Record**

Non In Place Updates have 2 length fields the first representing after image and the second for the before image

Log Record – Where is the data



Update To Varchar – Increasing Length

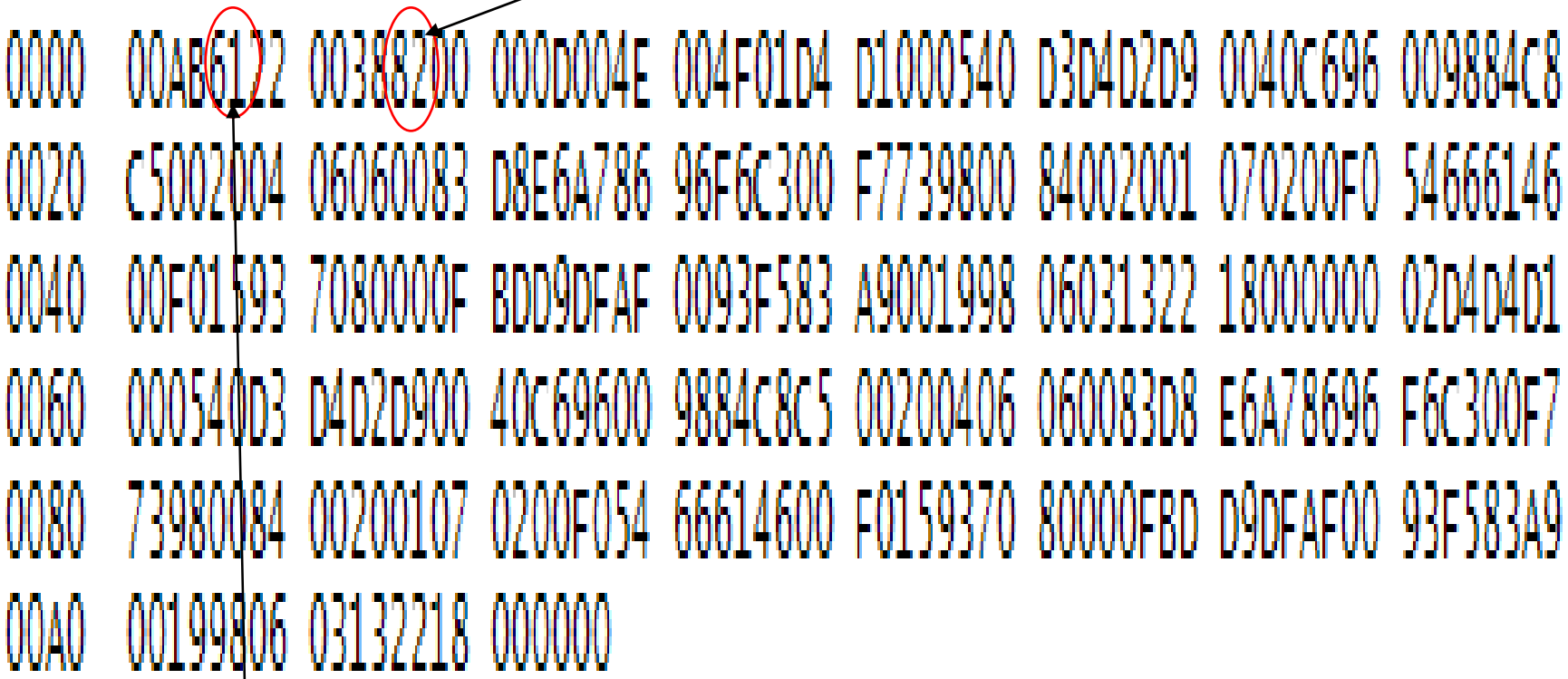
- **The data logged is from the first changed byte to the end of the row.**
- **The first Byte change is the second byte of the length field for FIRSTNME as the length changed from 1 to 5**
- **If there is not enough room on the page then a Pointer record will be written**
- **If there is enough room on the page then DB2 will move the new format of the row to a new location**
- **The RID will stay the same**

Third Update - Reduce the length

```
UPDATE DSN8810.EEMP
SET FIRSTNME = 'M'
WHERE FIRSTNME = 'MM'
```

Making a Varchar Shorter – What is logged

LGBFLGS2 Hex 82 is Binary 10000010



LGBFLGS Hex 61 is Binary 01100001

LGBFLGS- Meaning of Bitmap

B'10000000' YES IF UPDATE TO HASH ANCHOR
NO IF RECORD CHANGE.

B‘zXXzzzzz' TYPE OF SUB-OPERATION: (LGBSUBOP) XX =
00 IF IN-PLACE RECORD UPDATE OR HASH-ANCHOR UPDATE.
10 IF INSERT
01 IF DELETE
11 IF NON-IN PLACE REC UPDATE

B'01000000' YES IF RECORD INSERTION

B'00100000' YES IF RECORD DELETION

B'00010000' YES IF ID-MAP ENTRY ADDED OR DELETED FOR RECORD ISRT/DLET

B'00001100' TYPE OF DATA CAPTURE OPERATIONS

B'00001000' BIT =1, TABLE IS DEFINED FOR DATA CAPTURE.

B'00000100' BIT =1, RECURSIVE DATA CAPTURE OPERATION NOT DONE.

B'00000010' BIT =1, RI CAUSED REPLACE OR DELETE.

B'00000001' YES IF LOG RECORD IS WRITTEN ON BEHALF OF AN SQL UPDATE OPERATION.

YES AND LGBSUBOP = '10' - LOG RECORD REPRESENTS THE INSERTION OF AN OVERFLOW RECORD ON BEHALF OF AN SQL UPDATE OPERATION.

LGBFLGS2- Meaning of Bitmap

B'10000000'	1 IF PARTIAL IMAGE LOGGED DOES NOT INCLUDE PREFIX DIFFERENCES IN PGSLTH (DATA RECORD LENGTH).
B'01000000'	1 IF RECORD HAS COMPRESSED DATA.
B'00100000'	1 IF RECORD HAS EXPANDED FORMAT LIKE CDC LOG RECORDS.
B'00010000'	1 IF GROSS LOCK IS ON
B'00001111'	VARIATION NUMBER:

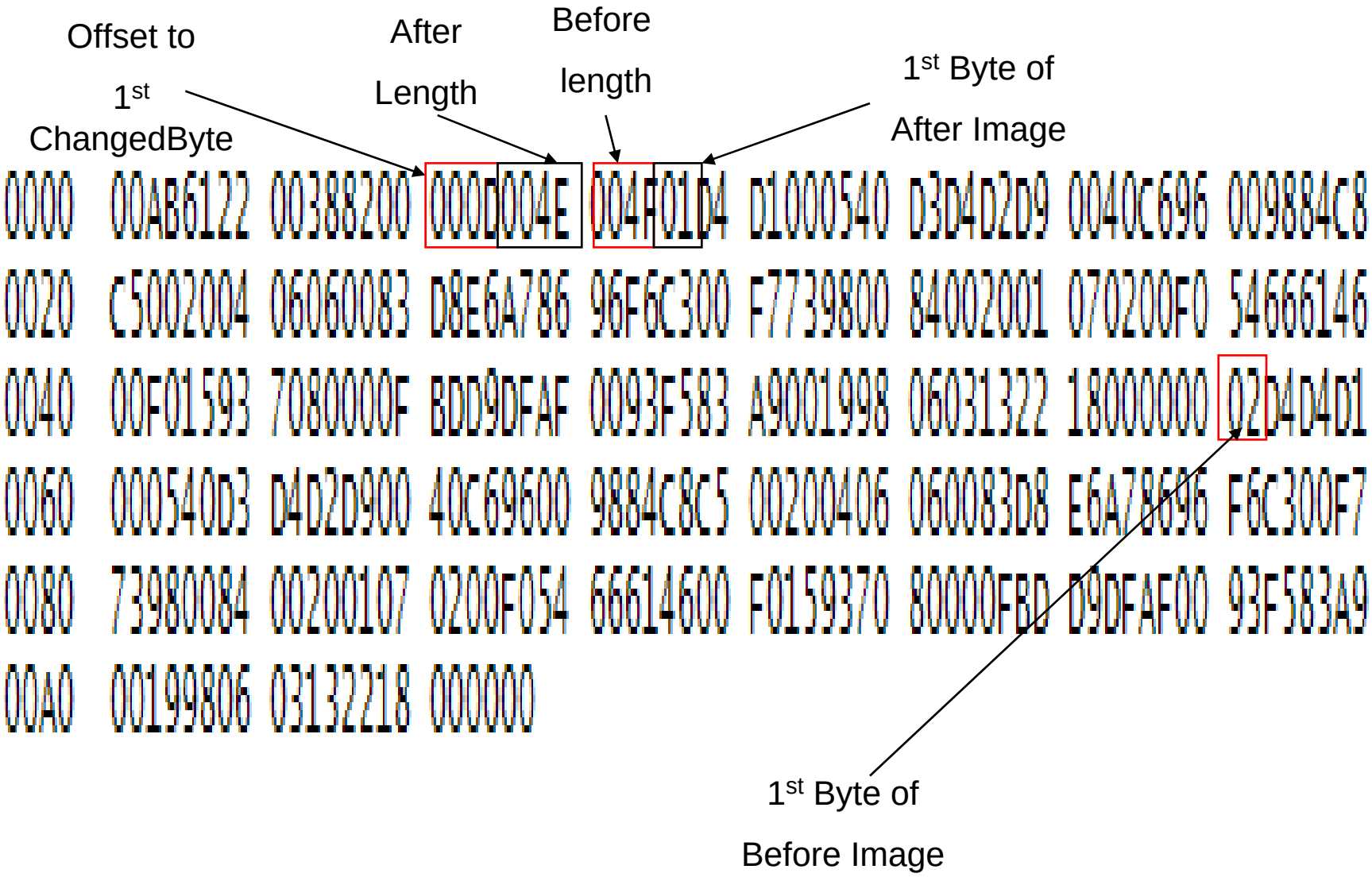
- 0000 INSERT OR DELETE LOG RECORD
- 0001 UPDATE VARIATION 1 LOG RECORD
- 0010 UPDATE VARIATION 2 LOG RECORD
- 0011 UPDATE VARIATION 3 LOG RECORD
- 0100 UPDATE VARIATION 4 LOG RECORD (EXCEPT 1ST LOG REC IN SERIES)
- 0101 UPDATE VARIATION 5 LOG RECORD (EXCEPT 1ST LOG REC IN SERIES)
- 0110 UPDATE VARIATION 6 LOG RECORD (EXCEPT 1ST LOG REC IN SERIES)
- 0111 UPDATE VARIATION 7 LOG RECORD (EXCEPT 1ST LOG REC IN SERIES)
- 1000 UPDATE VARIATION 8 LOG RECORD (EXCEPT 1ST LOG REC IN SERIES)
- 1001 UPDATE VARIATION 9 LOG REOCRD
- 1110 1ST LOG RECORD IN A VARIATION 4 OR 5 SERIES
- 1111 1ST LOG RECORD IN A VARIATION 6, 7 OR 8 SERIES

What Kind of Log Record ?

- **UNDO/REDO**
- **Non In Place Update**
- **Partial Image**
- **Variation 2 Update Record**

Non In Place Updates have 2 length fields the first representing after image and the second for the before image

Making a Varchar Shorter – What is logged



How Long Does Recovery take ?

```
DSNU000I    295 10:02:45.42 DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = FLETCHP.FLETCHPR
DSNU1044I   295 10:02:45.52 DSNUGTIS - PROCESSING SYSIN AS EBCDIC
DSNU050I    295 10:02:45.73 DSNUGUTC -  RECOVER TABLESPACE DSN8D81A.DSN8S81R DSNUM ALL TOCOPY
FLETCHP.PB1I.IC.DSN8D81A.DSN8S81R.G0003V00
DSNU515I    295 10:02:46.14 DSNUCBAL - THE IMAGE COPY DATA SET FLETCHP.PB1I.IC.DSN8D81A.DSN8S81R.G0003V00 WITH
DATE=20070921 AND TIME=134127
                    IS PARTICIPATING IN RECOVERY OF TABLESPACE DSN8D81A.DSN8S81R
DSNU504I    295 10:05:43.85 DSNUCBMD - MERGE STATISTICS FOR TABLESPACE DSN8D81A.DSN8S81R   -
                    NUMBER OF COPIES=1
                    NUMBER OF PAGES MERGED=138689
                    ELAPSED TIME=00:02:57
DSNU500I    295 10:05:46.94 DSNUCBDR - RECOVERY COMPLETE, ELAPSED TIME=00:03:01
DSNU050I    295 10:05:46.95 DSNUGUTC -  REBUILD INDEX(ALL) TABLESPACE DSN8D81A.DSN8S81R
DSNU718I  @PB1I 295 10:05:46.96 DSNUCINT - NO INDEXES FOUND FOR TABLESPACE 'DSN8D81A.DSN8S81R'
DSNU010I    295 10:05:47.00 DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=4
```

COBOL Programs to Update BRF Table

- **2 COBOL Programs**
  - 1 to Update a Fixed Length Column (TSTAMP)
  - 1 to update the first Varchar Column (FIRSTNME)

Keep the timings to compare later

SQL In Program to update TSTAMP Column

```
EXEC SQL
   DECLARE C1 CURSOR WITH HOLD FOR
   SELECT TSTAMP
   FROM DSN8810.EEMP
   FOR UPDATE OF TSTAMP
END-EXEC.
 EXEC SQL
    FETCH FROM C1
    INTO :W-TSTAMP
END-EXEC.
EXEC SQL
   UPDATE DSN8810.EEMP
   SET TSTAMP = CURRENT TIMESTAMP
   WHERE CURRENT OF C1
END-EXEC.
```

Program Timings for Updating TSTAMP Column

```
 PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV  PG   PAGE   SWAP    VIO
 PLFUPD1     00    239   2.74    .00   3.61   182M   0      0      0      0
NAME-FLETCHP            TOTAL CPU TIME=  2.74  TOTAL ELAPSED TIME=  3.62
```

0000500001 ROWS UPDATED

Program to Update Firstnme column

```
EXEC SQL
  DECLARE C1 CURSOR WITH HOLD FOR
  SELECT FIRSTNME
  FROM DSN8810.EEMP
  FOR UPDATE OF FIRSTNME
END-EXEC.

EXEC SQL
  OPEN C1
END-EXEC.
EXEC SQL
  FETCH FROM C1
  INTO :W-FIRSTNME
END-EXEC.
MOVE 'PAUL' TO W-FIRSTNME-TEXT.
MOVE 4 TO W-FIRSTNME-LEN.
EXEC SQL
  UPDATE DSN8810.EEMP
  SET FIRSTNME = :W-FIRSTNME
  WHERE CURRENT OF C1
END-EXEC.
```

Timings For Updating FIRSTNME

```
 PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV  PG   PAGE   SWAP    VIO
 PLFUPD2     00    219   2.73    .00   3.68   180M   0      0      0      0
NAME-FLETCHP            TOTAL CPU TIME=  2.73  TOTAL ELAPSED TIME=  3.68
```

00005500001 ROWS UPDATED

Next Steps

- **Recover to Image Copy**
- **Reorg to convert to RRF**
- **Image Copy**
- **UNLOAD**
- **DSNTIAUL**
- **UPDATES & Log Prints**
- **UPDATES With COBOL Programs**

Timings for Converting to RRF (Reorg)

```
 PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV  PG   PAGE   SWAP    VIO
 DSNUPROC    04  35409    .47    .00   3.14 41428K   0      0      0      0
NAME-FLETCHP             TOTAL CPU TIME=   .47  TOTAL ELAPSED TIME=  3.14
```

```
DSNU000I    296 14:51:43.84 DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = FLETCHP.FLETCHPR
DSNU1044I   296 14:51:43.87 DSNUGTIS - PROCESSING SYSIN AS EBCDIC
DSNU050I    296 14:51:43.89 DSNUGUTC -  REORG TABLESPACE DSN8D81A.DSN8S81R LOG NO SORTDATA SORTDEVT SYSDA SORTNUM 4
DSNU301I  @PB1I 296 14:51:43.90 DSNURFIT - KEYWORD 'SORTDATA' SPECIFIED AND/OR KEYWORD 'NOSYSREC' WAS SPECIFIED BUT NO
CLUSTERING INDEX EXISTS, KEYWORD IS IGNORED
DSNU252I    296 14:53:48.86 DSNURULD - UNLOAD PHASE STATISTICS - NUMBER OF RECORDS UNLOADED=5500001 FOR TABLESPACE
DSN8D81A.DSN8S81R
DSNU250I    296 14:53:48.86 DSNURULD - UNLOAD PHASE COMPLETE, ELAPSED TIME=00:00:42
DSNU304I  @PB1I 296 14:54:49.87 DSNURWT - (RE)LOAD PHASE STATISTICS - NUMBER OF RECORDS=0 FOR TABLE DSN8810.EEPA
DSNU304I  @PB1I 296 14:54:49.87 DSNURWT - (RE)LOAD PHASE STATISTICS - NUMBER OF RECORDS=0 FOR TABLE DSN8810.EPROJACT
DSNU304I  @PB1I 296 14:54:49.87 DSNURWT - (RE)LOAD PHASE STATISTICS - NUMBER OF RECORDS=0 FOR TABLE DSN8810.EACT
DSNU304I  @PB1I 296 14:54:49.87 DSNURWT - (RE)LOAD PHASE STATISTICS - NUMBER OF RECORDS=0 FOR TABLE DSN8810.EPROJ
DSNU304I  @PB1I 296 14:54:49.87 DSNURWT - (RE)LOAD PHASE STATISTICS - NUMBER OF RECORDS=5500001 FOR TABLE DSN8810.EEMP
DSNU304I  @PB1I 296 14:54:49.87 DSNURWT - (RE)LOAD PHASE STATISTICS - NUMBER OF RECORDS=0 FOR TABLE DSN8810.EDEPT
DSNU302I    296 14:54:50.06 DSNURILD - (RE)LOAD PHASE STATISTICS - NUMBER OF INPUT RECORDS PROCESSED=5500001
DSNU300I    296 14:54:50.06 DSNURILD - (RE)LOAD PHASE COMPLETE, ELAPSED TIME=00:01:00
DSNU381I  @PB1I 296 14:54:50.08 DSNUGSRX - TABLESPACE DSN8D81A.DSN8S81R IS IN COPY PENDING
DSNU010I    296 14:54:50.09 DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=4
```

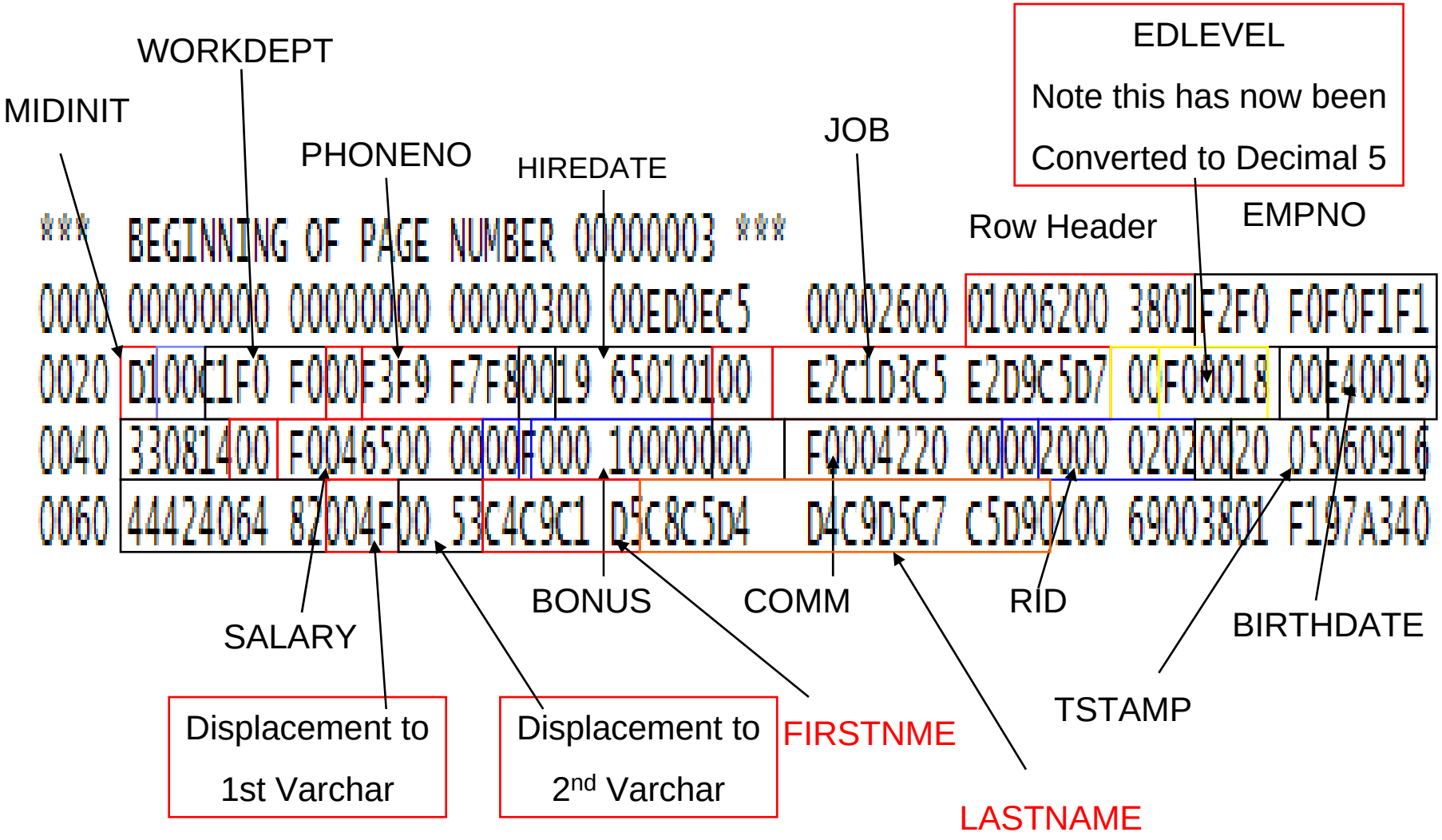
Now Run SQL to Check EEMP is in RRF

```
SELECT DISTINCT SUBSTR(A.CREATOR,1,8) AS CREATOR,
   A.NAME, B.FORMAT
FROM SYSIBM.SYSTABLES A, SYSIBM.SYSTABLEPART B
WHERE FORMAT <> ' '
AND A.DBNAME = B.DBNAME
AND A.TSNAME = B.TSNAME
AND A.CREATOR ¬= 'SYSIBM'
AND A.TYPE = 'T'
AND A.CREATOR = 'DSN8810'
```

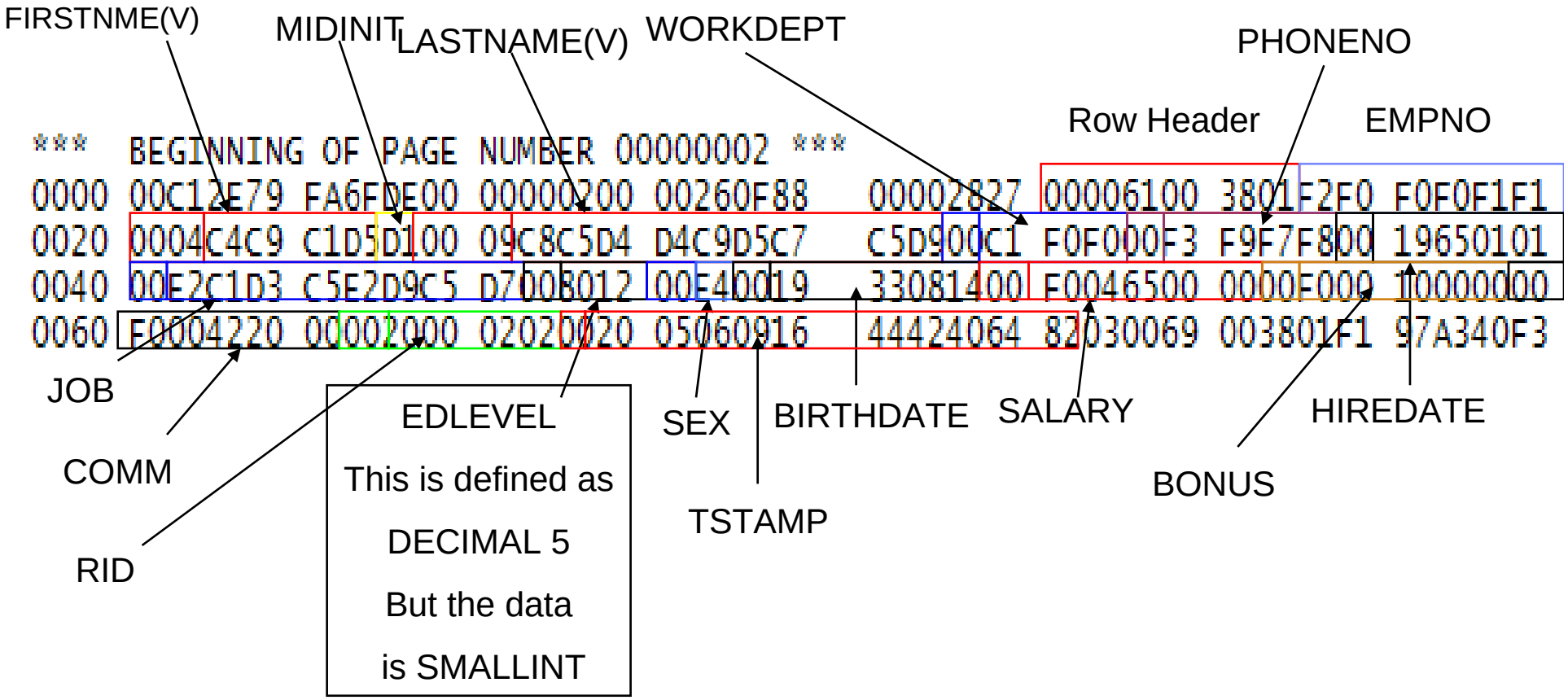
Output From SQL

CREATOR	NAME	FORMAT
*	*	*
--------	--------	------
DSN8810	EACT	R
DSN8810	EDEPT	R
DSN8810	EEMP	R
DSN8810	EEPA	R
DSN8810	EPROJ	R
DSN8810	EPROJACT	R

Print Of Tablespace after Reorg



Print of Basic Row Format Tablespace



Differences Between BRF Row and RRF Row

- **BRF has 2 Byte Length Followed by Data**
- **BRF column is placed where DDL stated**

- **RRF does NOT have 2 Byte length**
- **RRF has varying length data at end of row**
- **RRF has 2 byte displacement field for each column**

- **So Row Size WILL REMAIN THE SAME**

RRF Image Copy Timings

```
STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV  PG   PAGE   SWAP    VIO
COPY1    DSNUPROC    00  21444    .04    .00    .94  8206K   0      0      0      0
ENDED.  NAME-FLETCHP            TOTAL CPU TIME=   .04  TOTAL ELAPSED TIME=   .94
```

NUMBER OF PAGES=146039

AVERAGE PERCENT FREE SPACE PER PAGE = 5.67

PERCENT OF CHANGED PAGES = 0.01

ELAPSED TIME=00:00:53

Compare Image Copy Timings

	BRF	RRF
CPU	0.03	0.04
EXCP	20399	21444
ELAPSED	0.9	0.94
PAGES	138691	146039
%FREE SPACE PER PAGE	0.74	5.67

UNLOAD Utility Timings

PROCSTEP	RC	EXCP	CPU	SRB	CLOCK	SERV
DSNUPROC	00	28168	.61	.00	.77	53120K

TOTAL CPU TIME= .61 TOTAL ELAPSED TIME= .77

NUMBER OF RECORDS UNLOADED=0 FOR TABLE DSN8810.EDEPT

NUMBER OF RECORDS UNLOADED=5500001 FOR TABLE

NUMBER OF RECORDS UNLOADED=0 FOR TABLE DSN8810.EPROJ

NUMBER OF RECORDS UNLOADED=0 FOR TABLE DSN8810.EACT

NUMBER OF RECORDS UNLOADED=0 FOR TABLE

NUMBER OF RECORDS UNLOADED=0 FOR TABLE DSN8810.EEPA

NUMBER OF RECORDS UNLOADED=5500001 FOR TABLESPACE

Unload Utility Comparison

	BRF	RRF
EXCP	24172	28168
CPU	0.47	0.61
ELAPSED	1.04	0.77

Note: The format of the output from the UNLOAD is the same as it was when TABLESPACE was BRF

DSNTIAUL Timings

```
 PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV  PG   PAGE   SWAP    VIO
 DELET0      00      8    .00    .00    .00    782   0      0      0      0
LIMIT TO 2GB
 UNLD0       00  26662    .83    .00   1.12 49341K   0      0      0      0
NAME-FLETCHP             TOTAL CPU TIME=   .83  TOTAL ELAPSED TIME=  1.12
```

DSNT490I SAMPLE DATA UNLOAD PROGRAM

DSNT503I UNLOAD DATA SET SYSPUNCH RECORD LENGTH SET TO 80

DSNT504I UNLOAD DATA SET SYSPUNCH BLOCK SIZE SET TO 27920

DSNT503I UNLOAD DATA SET SYSREC00 RECORD LENGTH SET TO 134

DSNT504I UNLOAD DATA SET SYSREC00 BLOCK SIZE SET TO 27872

DSNT495I SUCCESSFUL UNLOAD 5500001 ROWS OF TABLE "DSN8810"."EEMP"

DSNTIAUL Comparison

	BRF	RRF
EXCP	22814	26662
CPU	0.80	0.83
ELAPSED	1.12	1.12

Update Statement

```
UPDATE DSN8810.EEMP
SET SALARY = SALARY + 1000
WHERE SALARY = 95652.58
```

What is Logged

```
0006AD81D3FB  TYPE( UNDO  REDO )  URID(0006AD81D33C)
              LRSN(C1648DE7EDB4)  DBID(0104)  OBID(0006)  PAGE(00000003)       09:0
              SUBTYPE(UPDATE IN-PLACE IN A DATA PAGE)  CLR(NO)
              PROCNAME(DSNIREPR)

     *LRH* 0048002F 06000001 0E800006 AD81D33C 0006AD81 D3CC0626 0006AD81 D3CCC164
           8DE7EDB4 0001
     *LG** 80010400 06000000 0300C164 8DCD8748 2D00
     0000  00100102 00388210 00326652 58565258
```

Is Anything Different between log for BRF and RRF

Basic Row Format

```
0000DAA1B03D  TYPE( UNDO  REDO )  URID(0000DAA1AF0A)
              LRSN(C139E7FB8096)  DBID(0104)  OBID(0006)  PAGE(00000002)       10:59
              SUBTYPE(UPDATE IN-PLACE IN A DATA PAGE)  CLR(NO)
              PROCNAME(DSNIREPR)

     *LRH* 0048003D 06000001 0E800000 DAA1AF0A 0000DAA1 B0000626 0000DAA1 B000C139
           E7FB8096 0001
     *LG** 80010400 06000000 0200C12E 79FA6FDE 2D00
     0000  0010 01 02 0038 82 00 004A6652 58565258
```

Length LGBFLGS ID OBID LGBFLGS2&3 DISPL NEW OLD

Reordered Row Format

```
0006AD81D3FB  TYPE( UNDO  REDO )  URID(0006AD81D33C)
              LRSN(C1648DE7EDB4)  DBID(0104)  OBID(0006)  PAGE(00000003)       09:0
              SUBTYPE(UPDATE IN-PLACE IN A DATA PAGE)  CLR(NO)
              PROCNAME(DSNIREPR)

     *LRH* 0048002F 06000001 0E800006 AD81D33C 0006AD81 D3CC0626 0006AD81 D3CCC164
           8DE7EDB4 0001
     *LG** 80010400 06000000 0300C164 8DCD8748 2D00
     0000  0010 01 02 0038 82 10 0032 6652 58565258
```

Update of Varchar Column

Colum	ColNum	Datatype	Length	Scale	Null
EMPNO	1	CHAR	6	0	N
FIRSTNME	2	VARCHAR	12	0	N
MIDINIT	3	CHAR	1	0	N
LASTNAME	4	VARCHAR	15	0	N
WORKDEPT	5	CHAR	3	0	Y
PHONENO	6	CHAR	4	0	Y
HIREDATE	7	DATE	4	0	Y
JOB	8	CHAR	8	0	Y
EDLEVEL	9	DECIMAL	5	0	Y
SEX	10	CHAR	1	0	Y
BIRTHDATE	11	DATE	4	0	Y
SALARY	12	DECIMAL	9	2	Y
BONUS	13	DECIMAL	9	2	Y
COMM	14	DECIMAL	9	2	Y
RID	15	CHAR	4	0	Y
TSTAMP	16	TIMESTMP	10	0	Y

First Update – Leave the length the same

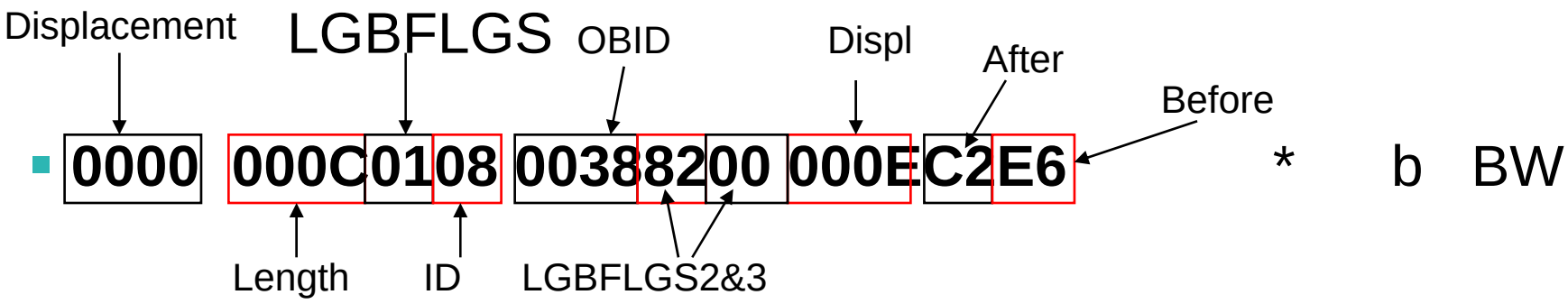
```
UPDATE DSN8810.EEMP
SET FIRSTNME = 'B'
WHERE FIRSTNME = 'W'
```

What Is Logged for Varchar Leaving Length Same

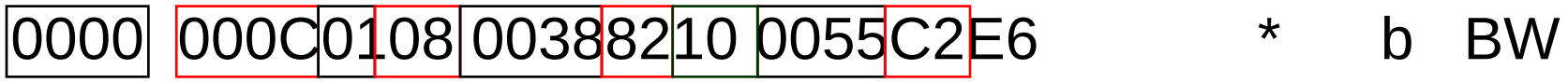
```
0006AE5B5709  TYPE( UNDO  REDO )  URID(0006AE5B564A)
              LRSN(C1675AC2B3EB)  DBID(0104)  OBID(0006)  PAGE(00000003)      14:32:36 07.299
              SUBTYPE(UPDATE IN-PLACE IN A DATA PAGE)  CLR(NO)
              PROCNAME(DSNIREPR)

    *LRH* 0044002F 06000001 0E800006 AE5B564A 0006AE5B 56DA0626 0006AE5B 56DAC167 *          $ ›  $     $ A
          5AC2B3EB 0001                                                          *!B
    *LG** 80010400 06000000 0300C164 8DE7EDB4 2D00                               *        A  X
    0000  000C0108 00388210 0055C2E6                                             *     b  BW
```

What Is Logged for Update keeping length same

Basic Row Format



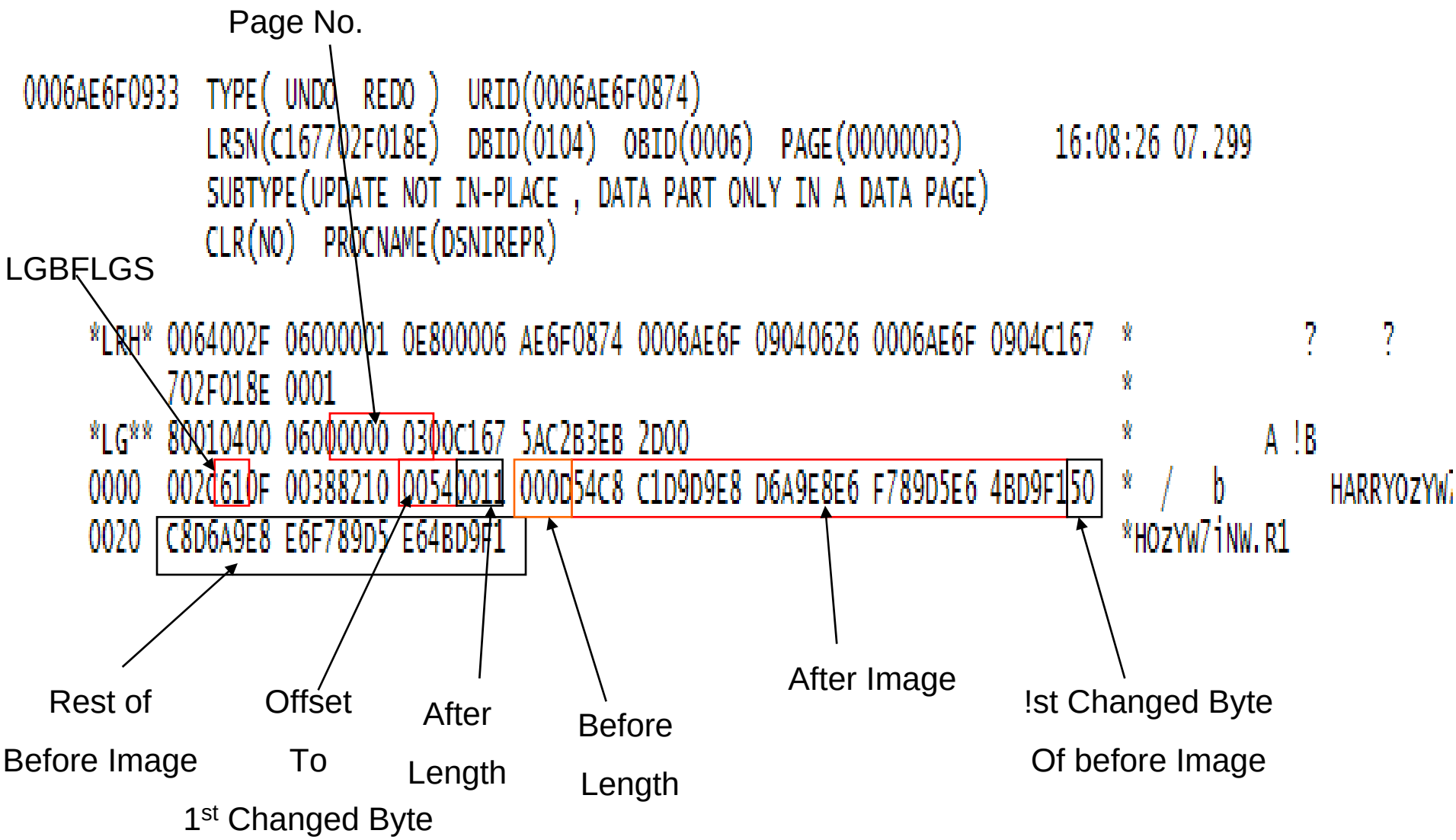
Reordered Row Format



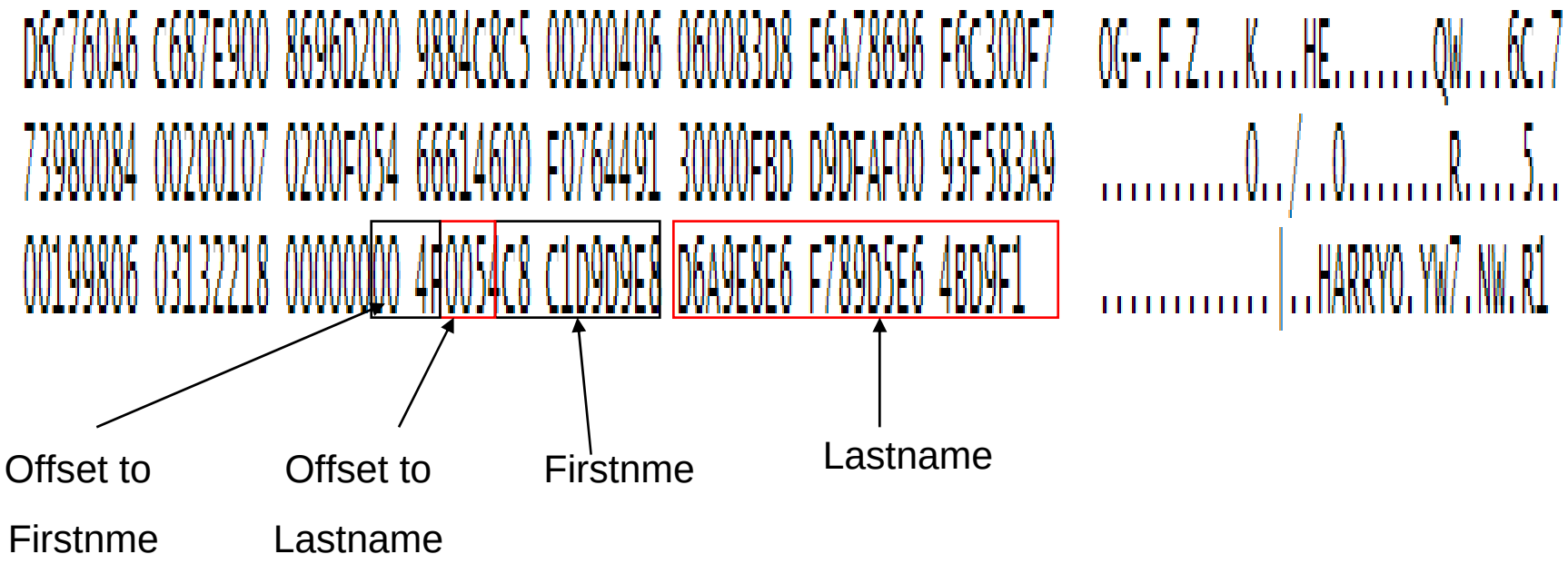
Second Update Increase the length

```
UPDATE DSN8810.EEMP
SET FIRSTNME = 'HARRY'
WHERE FIRSTNME = 'H'
```

What is logged after increased length



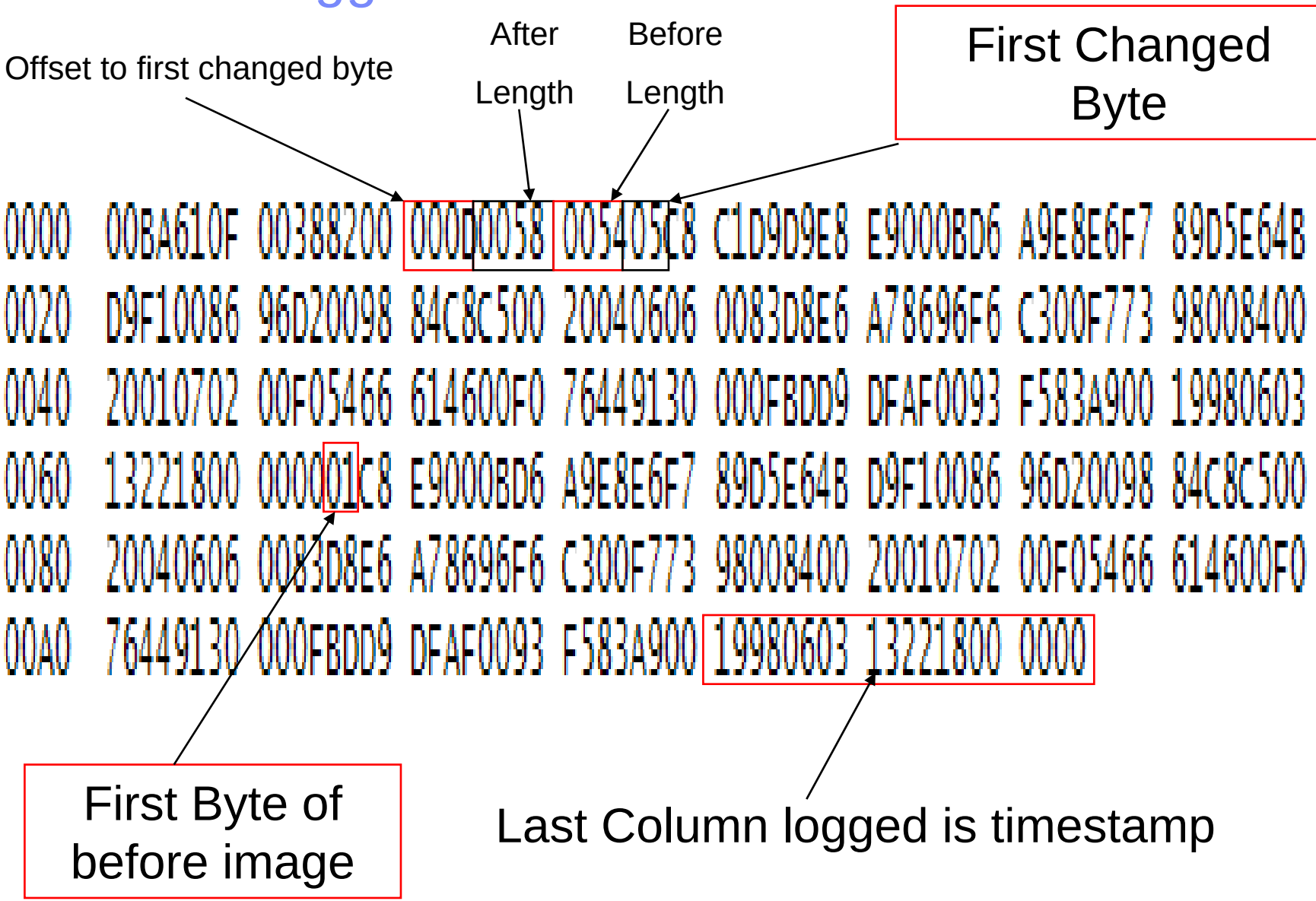
Print of Changed Row



Summary of what was logged

- **The length of Firstnme increased from 1 to 5 bytes**
- **The first changed byte was the offset to the second varchar**
- **Once again the data logged was from the first changed byte to the end of the row**
- **Total data logged was 13 bytes for the before image and 17 bytes for the after image**

What was logged when data was BRF



BRF V RRF Logging

Total Data Logged	BRF	RRF
	172	30

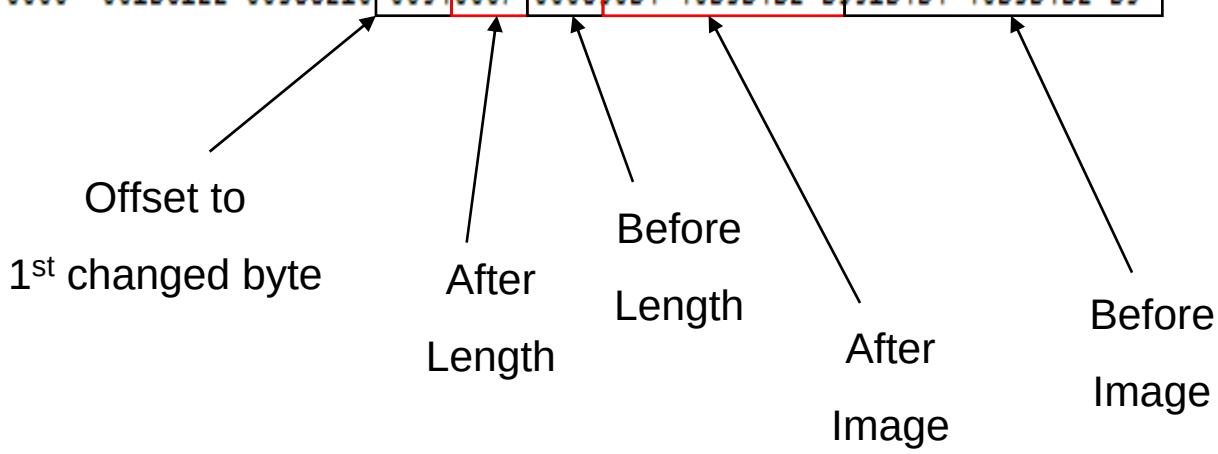
Moving the Varchars to the end of the row has saved 142 bytes of data logged

Third Update - Reduce the length

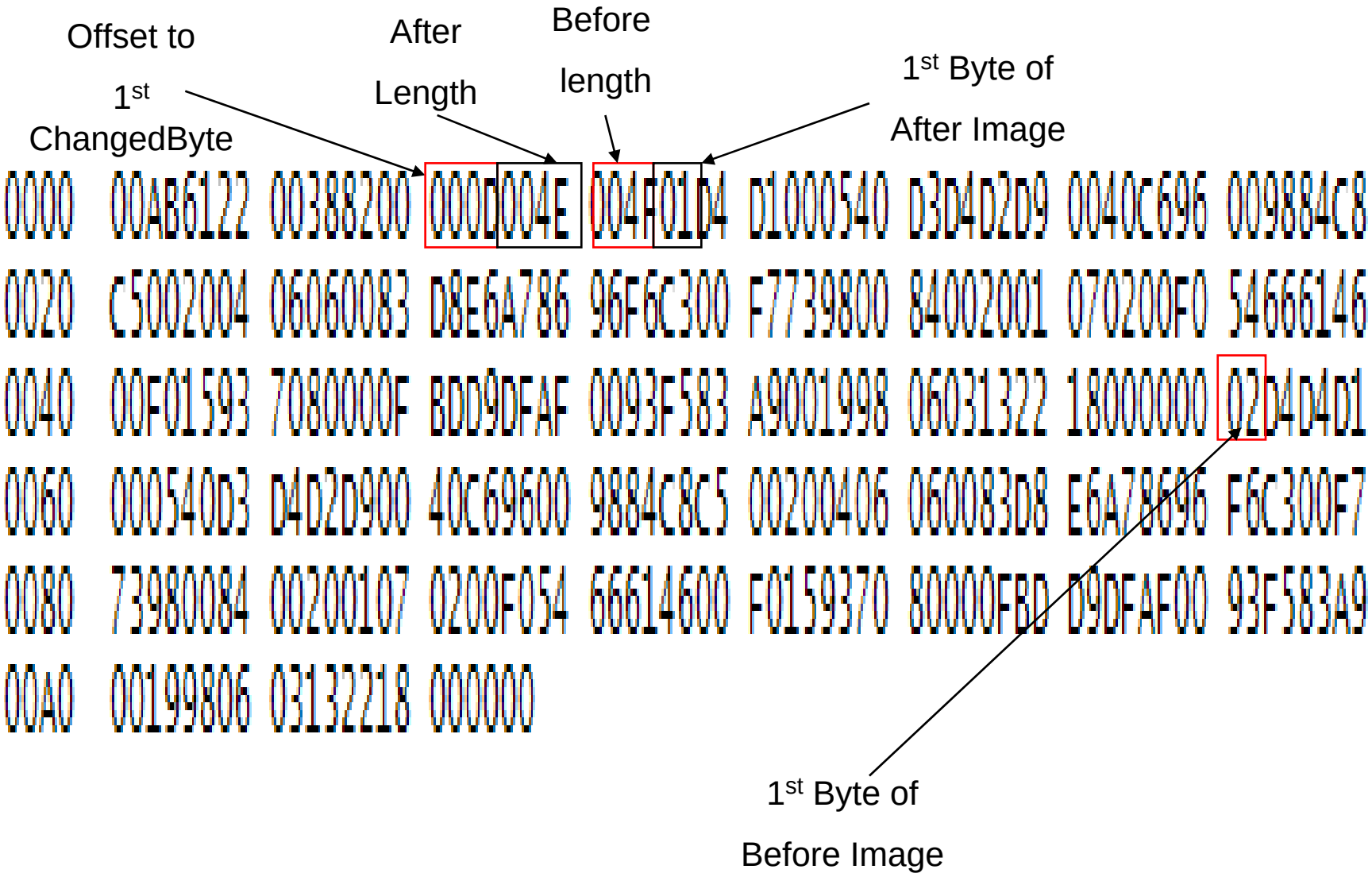
```
UPDATE DSN8810.EEMP
SET FIRSTNME = 'M'
WHERE FIRSTNME = 'MM'
```

What is logged for RRF after decrease in length

```
0006AEB26E87  TYPE( UNDO  REDO )  URID(0006AEB26DC8)
              LRSN(C16AE70C62ED)  DBID(0104)  OBID(0006)  PAGE(00000003)      09:16:12 07.302
              SUBTYPE(UPDATE NOT IN-PLACE , DATA PART ONLY IN A DATA PAGE)
              CLR(NO)  PROCNAME(DSNIREPR)

   *LRH* 0055002F 06000001 0E800006 AEB26DC8 0006AEB2 6E580626 0006AEB2 6E58C16A *            _H   >      > A
         E70C62ED 0001                                                          *X
   *LG** 80010400 06000000 0300C167 702F018E 2D00                               *         A
   0000  001D6122 00388210 00540007 00085 0D4 40D3D4D2 D951D4D4 40D3D4D2 D9      *  /   b      &M LMKR MM LMKR
```



What was logged for BRF



BRF V RRF Logging for reduced length

Total Data Logged	BRF	RRF
	157	15

RRF logged 142 bytes less data than BRF

COBOL Programs to Update

- **No REBIND needed after the conversion to RRF**
- **Both Plan and Package remain Valid**

SQL In Program to update TSTAMP Column

```
EXEC SQL
   DECLARE C1 CURSOR WITH HOLD FOR
   SELECT TSTAMP
   FROM DSN8810.EEMP
   FOR UPDATE OF TSTAMP
END-EXEC.
 EXEC SQL
    FETCH FROM C1
    INTO :W-TSTAMP
END-EXEC.
EXEC SQL
   UPDATE DSN8810.EEMP
   SET TSTAMP = CURRENT TIMESTAMP
   WHERE CURRENT OF C1
END-EXEC.
```

Timings for 1st COBOL Program BRF V RRF

BRF

```
 PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV  PG   PAGE   SWAP    VIO
 PLFUPD1     00    239   2.74    .00   3.61   182M   0      0      0      0
NAME-FLETCHP            TOTAL CPU TIME=  2.74  TOTAL ELAPSED TIME=  3.62
```

RRF

```
 PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV  PG   PAGE   SWAP    VIO
 PLFUPD1     00    266   2.64    .00   3.18   185M   0      0      0      0
NAME-FLETCHP            TOTAL CPU TIME=  2.64  TOTAL ELAPSED TIME=  3.18
```

Program to Update Firstnme column

```
EXEC SQL
  DECLARE C1 CURSOR WITH HOLD FOR
  SELECT FIRSTNME
  FROM DSN8810.EEMP
  FOR UPDATE OF FIRSTNME
END-EXEC.

EXEC SQL
  OPEN C1
END-EXEC.
EXEC SQL
  FETCH FROM C1
  INTO :W-FIRSTNME
END-EXEC.
MOVE 'PAUL' TO W-FIRSTNME-TEXT.
MOVE 4 TO W-FIRSTNME-LEN.
EXEC SQL
  UPDATE DSN8810.EEMP
  SET FIRSTNME = :W-FIRSTNME
  WHERE CURRENT OF C1
END-EXEC.
```

Timings For 2nd COBOL Program BRF V RRF

BRF

```
 PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV  PG   PAGE   SWAP    VIO
 PLFUPD2     00    219   2.73    .00   3.68   180M   0      0      0      0
NAME-FLETCHP             TOTAL CPU TIME=  2.73  TOTAL ELAPSED TIME=  3.68
```

RRF

```
STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV  PG   PAGE   SWAP    VIO
         PLFUPD1     00    266   2.50    .00   3.11   174M   0      0      0      0
ENDED.  NAME-FLETCHP             TOTAL CPU TIME=  2.50  TOTAL ELAPSED TIME=  3.11
```

Logging BRF V RRF

- **Logging stayed the same for fixed length columns**
- **Logging stayed the same for Varchars where length remained the same**
- **Logging decreased where the Varchar length either increased or decreased**

But are there any occasions when logging will increase?

Example of Increase In Logging in RRF

DDL For TABLE RRFTAB

CHARCOL	**CHAR(3)**
DECCOL	**DECIMAL(5,2)**
VARCOL1	**VARCHAR(200)**
VARCOL2	**VARCHAR(300)**
VARCOL3	**VARCHAR(20)**
VARCOL4	**VARCHAR(10)**

Physical Formats

BRF

CHARCOL	DECCOL	LL	VARCOL1	LL	VARCOL2	LL	VARCOL3

RRF

CHARCOL	DECCOL	OFF1	OFF2	OFF3	VARCOL1	VARCOL2	VARCOL3

Update Of RRFTAB

UPDATE RRFTAB

SET VARCOL2 = ‘PAUL FLETCHER’

WHERE VARCOL2 = ‘PAUL’

So What Will Be Logged For This Update?

What will be logged

BRF

(------Data Logged ----)

RRF

(----------Data Logged----------)

Examples of Creating Test Data

- **Extract from live with DSNTIAUL**
- **Load into test using LOAD Utility**

- **Extract from live using UNLOAD**
- **Load into test using Load Utility**

- **Extract from live using Reorg Unload**
- **Load into test using Load Utility**

- **Extract from live using DSN1COPY**
- **Load into test using DSN1COPY with XLAT**

- **Extract from live using home grown program**
- **Load into test using home grown program**

Will these work if Live is RRF and Test is BRF

- **Extract from live with DSNTIAUL**
- **Load into test using LOAD Utility**

- **Extract from live using UNLOAD**
- **Load into test using Load Utility**

- **Extract from live using Reorg Unload**
- **Load into test using Load Utility**

- **Extract from live using DSN1COPY**
- **Load into test using DSN1COPY with XLAT**

- **Extract from live using home grown program**
- **Load into test using home grown program**

It Depends

Summary

- **RRF puts all Varchars at the end of the row**
- **Less data may be logged**
- **Any programs using Varchars may speed up**
- **No Rebinds needed**
- **You get RRF after LOAD REPLACE or REORG**
- **If Data Capture Changes is Defined for this table then the whole row is logged so RRF will still log the same amount of data**