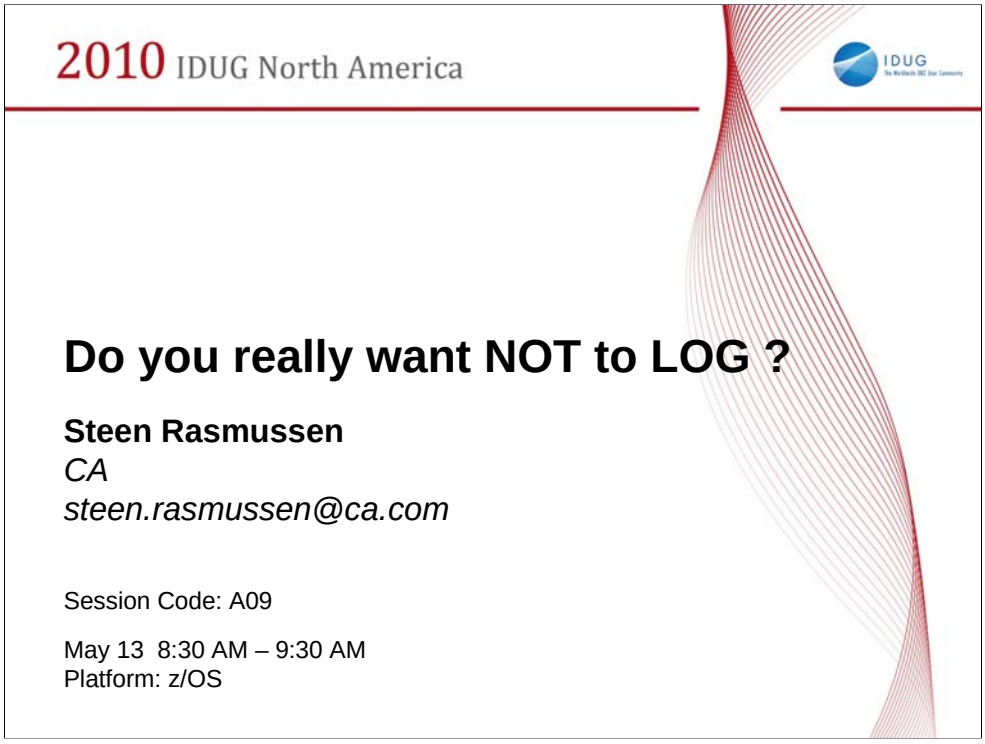


The attendee will get an A-Z walk through of the DB2 9 NOT LOGGED tablespace feature and what the impact is for applications as well as for backup / recovery and change propagation.

The presentation will also dig into the performance impact for SQL statements when NOT LOGGED tablespaces are referenced compared to LOGGED tablespaces.

Agenda

- Syntax changes implementing NOT LOGGED tablespaces
- Application and SQL impact when dealing with NOT LOGGED tablespaces
- Backup and Recovery considerations
- Performance case study
- Recommendations based on usage and performance

During this session we will look at the DDL changes, the requirements for using NOT LOGGED tablespaces as well as the limitations.

We will spend quite some time looking at the implications NOT LOGGED will have for applications and what to watch out for.

We will also cover a performance case study to see the performance impact of using NOT LOGGED – is there any ?

Finally a look into something you really should understand – and perhaps worry about when using NOT LOGGED – recoverability.

Logged / NOT logged characteristics

- Used in CREATE and ALTER statement
- LOGGED synonym with LOG YES
  (used to be for LOB's **only** up to 1GB)
- NOT LOGGED synonym with LOG NO
- Specified for the BASE tablespace
  - Every base table in the tablespace adopts attribute
  - Every index associated to a base table adopts attribute
  - Can also be specified for LOB tablespace
- If base tablespace altered to be NOT LOGGED -> LOB tablespace has to be NOT LOGGED too

A tablespace can be marked as NOT LOGGED on either a CREATE or ALTER statement.

In the past only LOB Auxiliary tablespace s could be marked as being not logged, but with DB2 9 both LOG YES/NO and NOT LOGGED are synonyms.

Please note that the NOT LOGGED attribute is specified for the tablespace – meaning all tables in the tablespace will be logged / not logged.

In the past the base table was logged and the LOB tablespace could be marked as not being logged. From now on, the logging parameter must be the same for both the base object and LOB object.

Logged / NOT logged characteristics

- Can not be specified for XML auxiliary tablespaces and XML indexes – base tablespace attribute is adopted
- Alter base -> XML objects inherit and are ***linked*** (please see next page)
- NOT LOGGED will not create REDO and UNDO log-records
- Control information for base tables still logged
- System pages and auxiliary indexes for LOB's are logged
- Data Capture Changes for tables residing in NOT LOGGED tablespaces cannot be specified

Unlike LOB objects where LOGGED / NOT LOGGED (or LOG YES / NO), can be specified, the logging attribute cannot be specified for XML objects. The XML objects will adopt the base table attribute.

The consequence og XML objects inheriting the logging attribute is : when a base tablespace is altered, the XML object is LINKED - please see the details on the next slide.

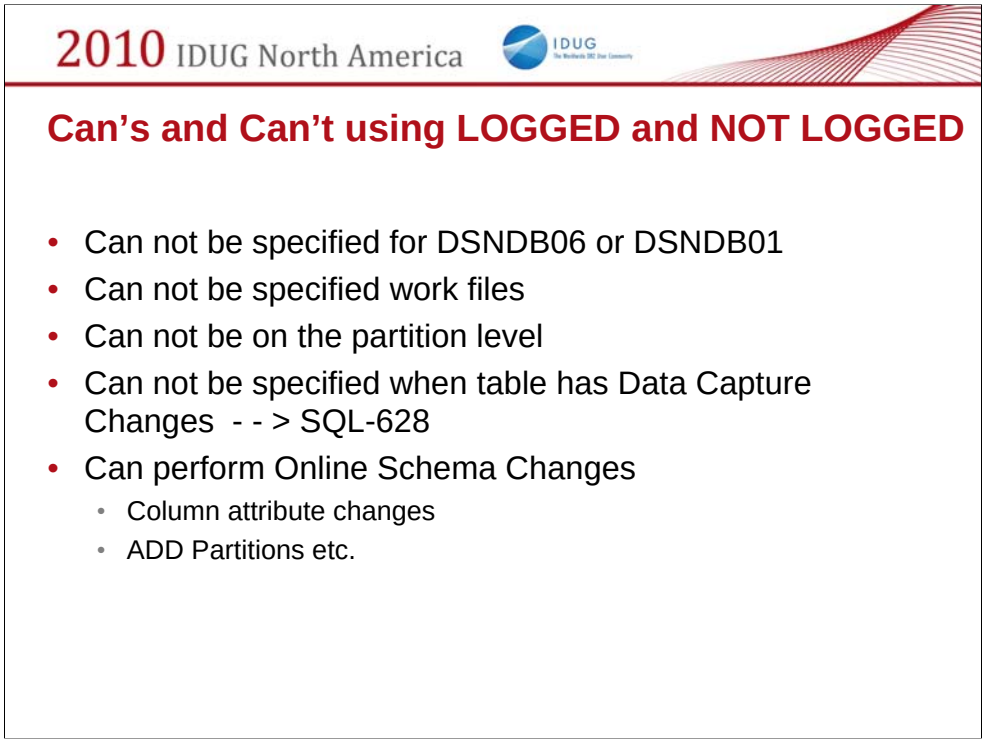
Logging is not performed for table and index updates/deletes/inserts when NOT LOGGED is active. All other information is still logged like control information, auxiliary indexes for LOB's, system pages for Online Schema Evolution etc.

Finally – Data Capture Changes cannot be enabled for a table when NOT LOGGED is defined for a tablespace.

LOGGED and LINK

- DB2 9
  - If LOB is LOGGED and base tablespace changed to NOT LOGGED – auxiliary tablespace implicitly altered to be NOT LOGGED – and ***LINKED***
    - *Can be seen in SYSIBM.SYSTABLESPACE.LOG(X)*
    - Dissolved when base altered back to logged
    - So LOB tablespace can be NOT LOGGED while base tablespace is LOGGED
  - If LOB altered to NOT LOGGED – it will remain so after base altered to LOGGED

The linked attribute can be viewed in SYSTABLESPACE column LOG. If the value is “X” – the LOB or XML tablespace is linked



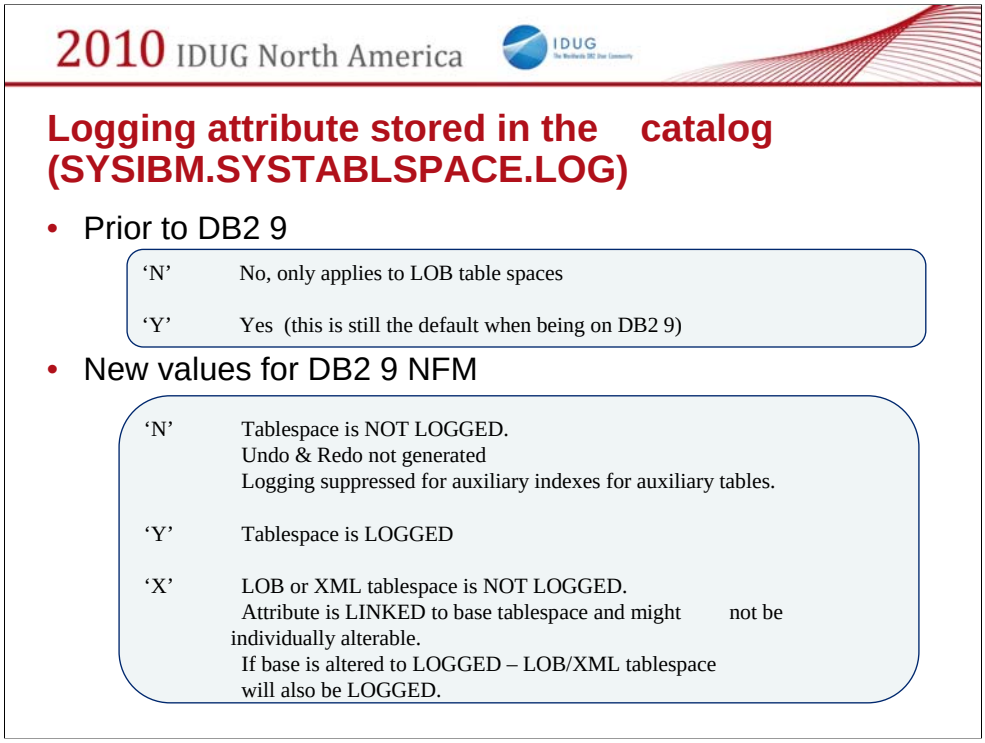
It is not possible to specify NOT LOGGED / LOG NO for all tablespace types.

The catalog and directory always have to be logged and so do work files.

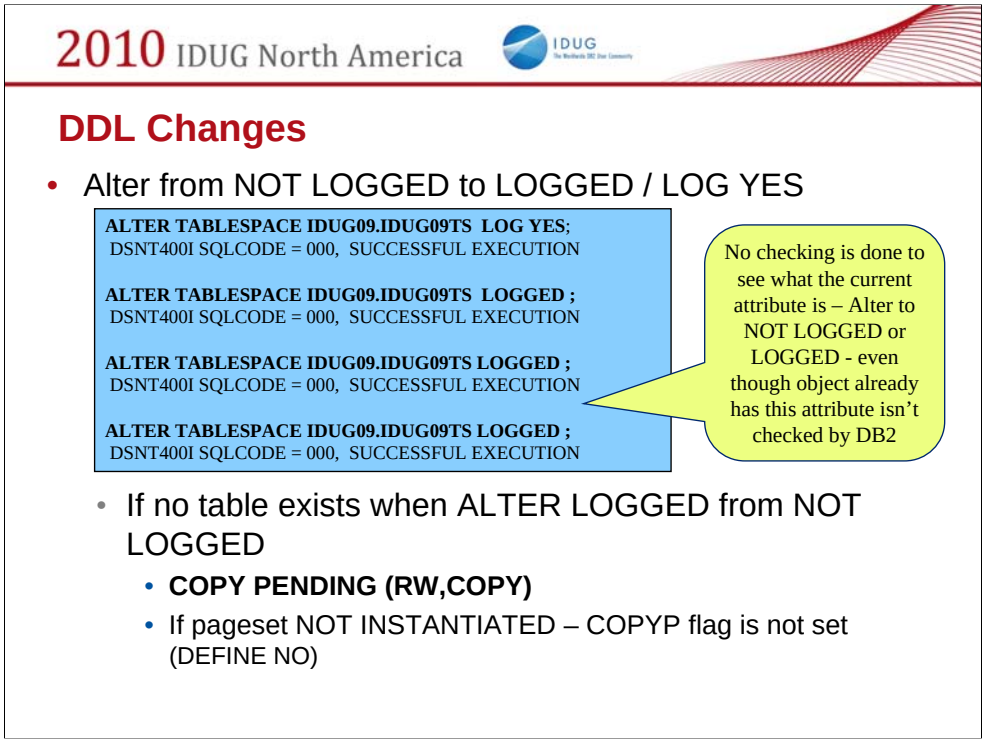
The LOGGED / NOT LOGGED attribute only works on the tablespace level, so all partitions in a partitioned tablespace will have the same parameter.

Due to the nature of DATA CAPTURE CHANGES, this parameter is mutually exclusive with NOT LOGGED and it is necessary to alter the table(s) to be DATA CAPTURE NONE prior to altering the tablespace to be NOT LOGGED.

There are no restrictions in terms of executing Online Schema Changes (unlike when Data Capture Changes is enabled where DB2 V8 and DB2 9 online schema changes isn’t possible.



As already mentioned, to verify if a tablespace is logged or not logged, this can be viewed in the catalog table SYSTABLESPACE and column=LOG



The SQL statement to use in order to turn LOGGING on/off is very simple. Even turning off logging when it's already is turned off is possible – no SQL WARNING issued.

When a tablespace is being altered to be LOGGED from NOT LOGGED, the status of the tablespace depends on a couple of things.

1) if the tablespace was created with DEFINE NO and hasn't been instantiated, the tablespace status remains unchanged.
2) If the tablespace has been instantiated, the tablespace status is set to COPY PENDING – even if no tables exist in the tablespace.

DDL Changes

- Remove LOGGING is very easy too
  - DB2 does NOT change the status of the pageset until first update
  - No COPYP setting (do you want an image copy?)
    - Start RO prior to turning off logging – and COPY !!!!!

```
ALTER TABLESPACE IDUG09.IDUG09TS NOT LOGGED ;
DSNT400I SQLCODE = 000,  SUCCESSFUL EXECUTION

SELECT SUBSTR(DSNAME,1,18),ICTYPE,STYPE,TIMESTAMP
 FROM    SYSIBM.SYSCOPY WHERE DBNAME='IDUG09' ;

#1                 ICTYPE STYPE TIMESTAMP
IDUG09.IDUG09TS    A      O     2008-11-04-17.03.12.083646
IDUG09.IDUG09TS    A      L     2008-11-04-16.45.51.131415
IDUG09.IDUG09TS    A      O     2008-11-04-16.40.13.330968
IDUG09.IDUG09TS    A      L     2008-11-04-16.36.41.908902
IDUG09.IDUG09TS    A      O     2008-11-04-16.36.41.904963
IDUG09.IDUG09TS    A      L     2008-11-04-16.36.41.900178
IDUG09.IDUG09TS    A      O     2008-11-04-16.36.41.866235
IDUG09.IDUG09TS    C      L     2008-11-04-16.31.34.520037
```

As easy as it is to change a tablespace to be LOGGED – it is as simple to make it NOT LOGGED.

Unlike a potential change to the (DBET) tablespace status when logging is activated, turning off logging has no effect on DBET.

Even though there is no change to the tablespace status, it is highly recommended to take an image copy right away (more about the reasoning later).

Note that the change of logging attribute (as well as the initial attribute) is being recorded in SYSIBM.SYSCOPY (more about this on the next page).

DDL Changes

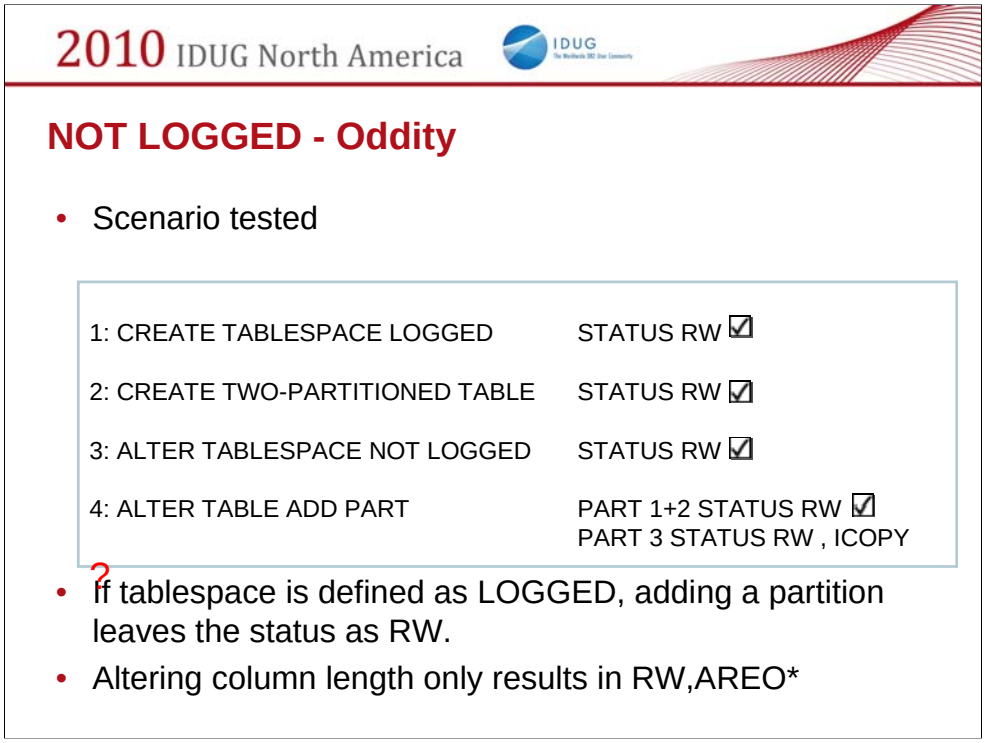
- SYSCOPY can grow quite a lot
  - All create's registered in SYSCOPY
  - Alter of LOGGING reflected
  - (more SYSCOPY information covered later)

```
SELECT SUBSTR(DSNAME,1,18),ICTYPE,STYPE,TIMESTAMP
FROM   SYSIBM.SYSCOPY   WHERE DBNAME='IDUG09' ;

#1                 ICTYPE STYPE TIMESTAMP
IDUG09.IDUG09TS    A      L     2008-11-04-16.45.51.131415
IDUG09.IDUG09TS    A      O     2008-11-04-16.40.13.330968
IDUG09.IDUG09TS    A      L     2008-11-04-16.36.41.908902
IDUG09.IDUG09TS    A      O     2008-11-04-16.36.41.904963
IDUG09.IDUG09TS    A      L     2008-11-04-16.36.41.900178
IDUG09.IDUG09TS    A      O     2008-11-04-16.36.41.866235
IDUG09.IDUG09TS    C      L     2008-11-04-16.31.34.520037
```

Something new has happened to SYSCOPY – when a tablespace is created it is also inserted into SYSCOPY. If you have a lot of CREATE activity going on – please consider expanding this catalog tablespace.

Just like some Online Schema Changes are reflected in SYSCOPY – so is turning LOGGING on and off.

We will cover SYSCOPY later too in order to get a more detailed picture of this "repository".



We have covered the status for tablespaces going through changing the logging attribute. This scenario is puzzling me a bit:

1)A tablespace is created – status is RW

2)A two-partitioned TCP is created – status still RW

3)The tablespace is altered to be NOT logged – status still RW

4)A new partition is added – and for some reason the original partitions remain RW while the new partition is placed in ICOPY status (informational image copy status)

Perhaps this is due to the fact that the new partition is in a NON-RECOVERABLE state ?

Performance Considerations

- Ever since NOT LOGGED was introduced for base tablespaces – IBM has stressed not to use NOT LOGGED to increase application performance
  - Use if DB2 logging is a real problem due to MASSIVE insert / update / delete activity for a table
    - One example is REFRESH of MQT
  - If it's easy to re-establish the starting point (LOAD REPLACE or EMPTY tablespace to start all over)
- Let's verify this statement

Some DB2 users are appreciating the fact that they can mark a tablespace not logged in order to save CPU for large amounts of insert operations etc.

However – IBM has (from day one) mentioned that the only reason to use NOT LOGGED is if the LOG CPU overhead adds more than 5% (ROT found in a performance guide).

IBM also states to only use NOT LOGGED if it's easy to establish the starting point again (like LOAD REPLACE, TRUNCATE, DELETE FROM).

Let's have a closer look to see if turning off logging really does save the application some DB2 CPU time.

Performance benchmarks

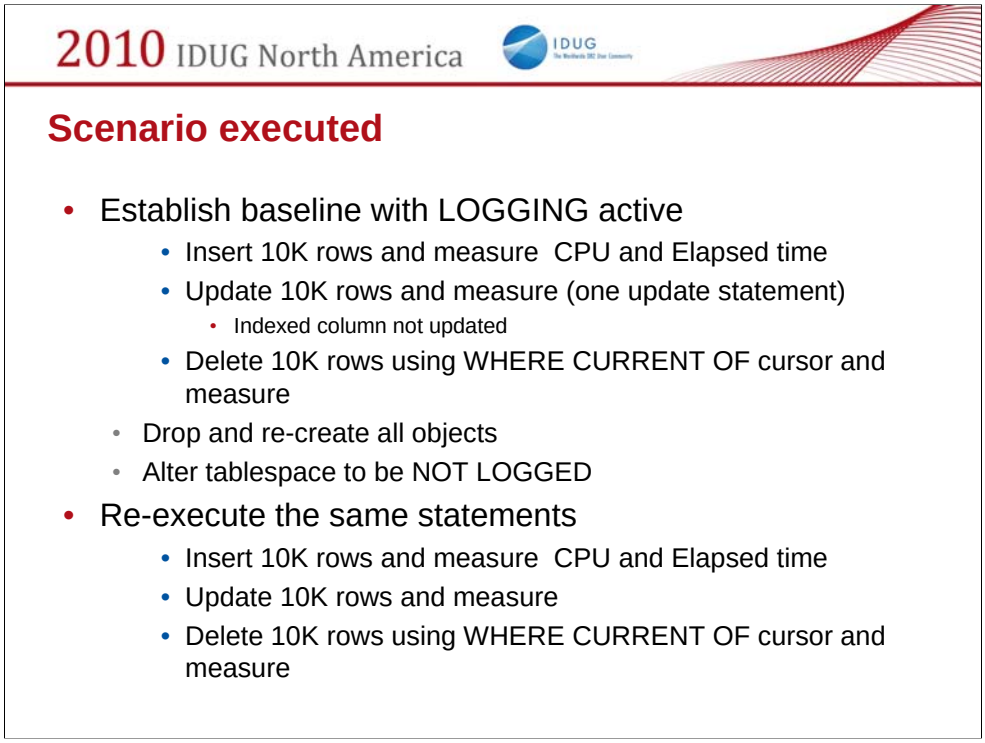
- DDL used throughout the executed benchmarks

```
CREATE  TABLESPACE IDUG09S8
      IN IDUG09
      USING STOGROUP SYSDEFLT
        FREEPAGE 0
        PCTFREE 5
        BUFFERPOOL BP0
        LOCKSIZE ANY
        CLOSE YES
        SEGSIZE 04
        LOCKMAX SYSTEM
        CCSID EBCDIC;
```

```
CREATE  TABLE RASST02.IDUGTB8
      ( C1 CHAR ( 10 )
      , C2 INTEGER
      , C3 DATE
      , C4 CHAR ( 11 )
      )
      IN IDUG09.IDUG09S8 ;


CREATE  INDEX RASST02.IDUGIX8
          ON      RASST02.IDUGTB8
      ( C1 ASC)
      USING STOGROUP SYSDEFLT
        FREEPAGE 0
        PCTFREE 10
        BUFFERPOOL BP0
        CLOSE YES    ;
```

- Data Capture Changes not enabled

The DDL listed were used to benchmark the difference from using LOGGED and NOT LOGGED tablespaces for a number of SQL operations.



The methodology used for the benchmarks were:

1)Create the tablespace LOGGED

2)Insert 10,000 rows and measure primarily CPU time

3)Update 10,000 rows in one SQL statement and measure CPU time

4)Delete 10,000 rows and measure CPU time

5)Re-create the object using NOT LOGGED

6)Re-execute the exact same statements

Scenario executed

- Only focus on application performance – not DB2 system performance
- Logging overhead and DASD consumption not monitored
- Elapsed time not monitored
- Object stopped and started between every execution
- Same scenario executed dozens of times and average calculated
- Non busy LPAR and DB2 system

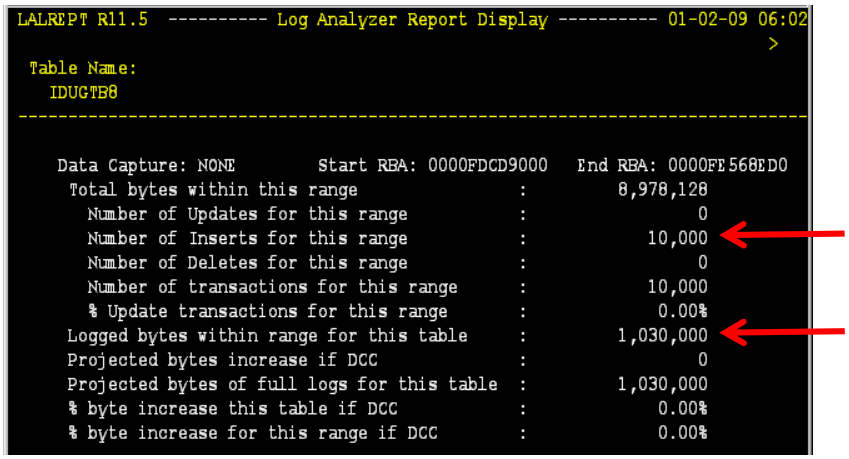
The outlined scenario was created in order to purely focus on the application performance – not DB2 logging or DB2 systems performance.

All tests were executed on a non busy LPAR and DB2 subsystem in order to eliminate that factor.

Each scenario were executed many times and the average CPU consumption calculated.

Logging for 10,000 Inserts

- Log-records and logged bytes for 10K Inserts



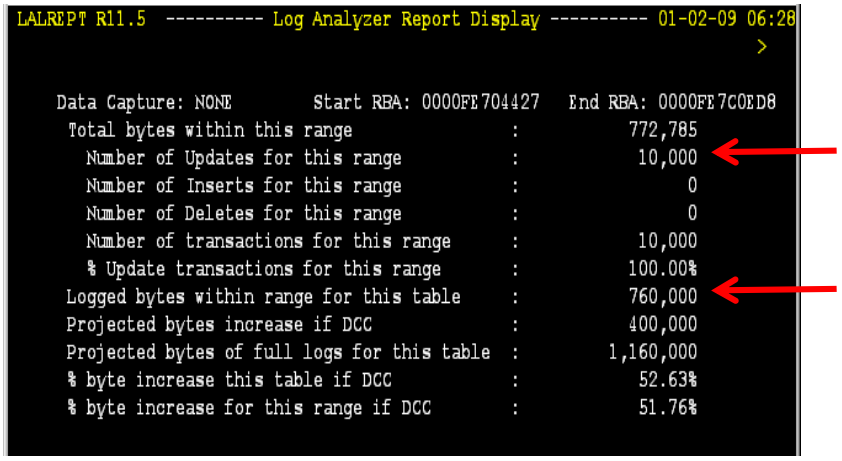
```
LALREPT R11.5  ---------- Log Analyzer Report Display ---------- 01-02-09 06:02
                                                                          >

 Table Name:
   IDUGTB8
--------------------------------------------------------------------------------

    Data Capture: NONE        Start RBA: 0000FDCD9000   End RBA: 0000FE568ED0
     Total bytes within this range                :        8,978,128
       Number of Updates for this range           :                0
       Number of Inserts for this range           :           10,000
       Number of Deletes for this range           :                0
       Number of transactions for this range      :           10,000
       % Update transactions for this range       :            0.00%
     Logged bytes within range for this table     :        1,030,000
     Projected bytes increase if DCC              :                0
     Projected bytes of full logs for this table  :        1,030,000
     % byte increase this table if DCC            :            0.00%
     % byte increase for this range if DCC        :            0.00%
```

This screen shot illustrates the number of INSERTS executed while the tablespace was in a LOGGED state.

We can also see that 1M bytes were logged during these executions.

Logging for 10,000 Updates

- Log-records and logged byte for 10K Updates



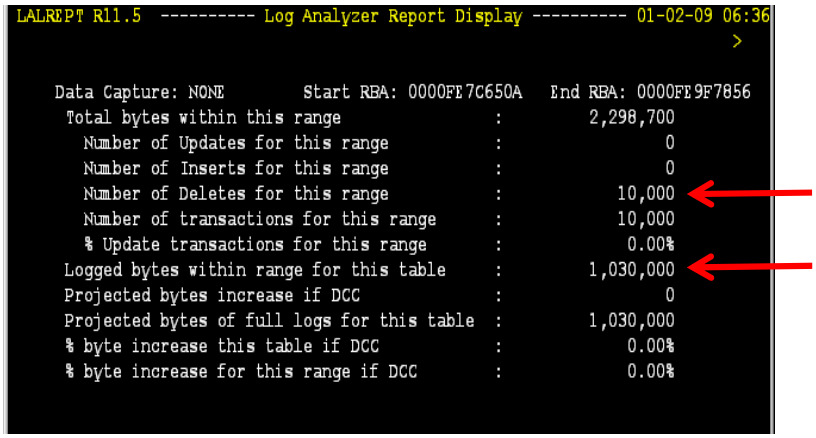
Logging report for the 10,000 updates for the LOGGED tablespace.

We can also verify Data Capture Changes were not enabled.

The reason for using Data Capture None was to verify the minimum logging done by DB2 for this statement for a LOGGED tablespace.

Logging for 10,000 Deletes

- Log-records and logged byte for 10K Deletes



Finally the logging report for 10,000 deletes for the LOGGED tablespace

Performance Comparison

	LOGGED CPU	NOT LOGGED CPU
10,000 INSERT	3.249	3.194
10,000 UPDATE	0.039	0.038
10,000 DELETE	0.766	0.746

According to my benchmarks – the IBM statement is very valid !

Don’t turn off logging to save CPU for the application SQL

The few benchmarks done proves IBM’s statement to be valid.

The CPU difference between using LOGGED and NOT LOGGED for the application is hardly worth mentioning.

Not Logged – is this really true ?

- Tablespace has LOGGED activated
  - Insert ‘1234567890’ into column C1
  - Update C1 to be ‘7777777777’
  - Delete the row
  - Execute DSN1LOGP
- Alter tablespace to be NOT LOGGED
  - Insert ‘1234567890’ into column C1
  - Update C1 to be ‘7777777777’
  - Delete the row
  - Execute DSN1LOGP
- Compare DSN1LOGP executions

Let’s have a closer look into if any logging is done for NOT LOGGED tablespaces.

The scenario outlined inserts one row, then updates one column and deletes the row. The same scenario was executed for both LOGGED and NOT LOGGED tablespace and DSN1LOGP is executed for both scenarios specifying the DBID, PSID to have a closer look at the real log content.

Now we can compare the DSN1LOGP content to see the real difference

INSERT LOGGED

- Log content formatted

```
URID: 00011DDBA2BE   Member         :
LRSN: C39EDE4A0CFB   Primary Auth-id: RASST02      Plan name      : RCUU1150
Date: 01-19-09       Correlation-id : RASST02      Connection-id  : DB2CALL
Time: 17:40:26.30    URID Status    : Committed    Connection Type:
TSO/Batch
------------------------------------------------------------------------------
-
  Table RASST02.IDUGTB8               Database: IDUG09    Tablespace:
IDUG09S

  Insert  RID: 0000000262  Log RBA: 00011DDBA34E  Log LRSN: C39EDE4A0CFB

          C1          C2        C3        C4
          ----------  --------  --------  --------

          1234567890  **NULL**  **NULL**  **NULL**
```

We can see result from the INSERTS is the entire row is logged for the LOGGED tablespace.

DSN1LOGP LOGGED / NOT LOGGED for INSERT

```
DSN1212I DSN1LGRD FIRST LOG  RBA ENCOUNTERED 00011DDBA34E

00011DDBA34E  TYPE( UNDO  REDO )  URID(00011DDBA2BE)
              LRSN(C39EDE4A0CFB)  DBID(0121)  OBID(0017)  PAGE(00000002)         17:40:26 09.019
              SUBTYPE(INSERT IN A DATA PAGE)  CLR(NO)  PROCNAME(DSNISGRT)

     *LRH* 00670090 06000001 0E800001 1DDBA2BE 00011DDB A2BE0626 00011DDB A2BEC3 9E *            s     s    s C
           DE4A0CFB 0000                                                            * ¢
     *LG** 88012100 17000000 02000001 1DD9CE9D 4C00                                 *h            R  <
     0000  002F4062 00181010 00002700 186200F1 F2F3F4F5 F6F7F8F9 F0FF0000 0000FF 00 *              1234567890
     0020  000000FF 00000000 00000000 000000                                        *

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

 ALL URIDS     === YOU MAY SPECIFY URID(XXXXXXXXXXX)
 ALL LUWIDS    === YOU MAY SPECIFY LUWID(NNNNNNNN.LLLLLLLL.XXXXXXXXXXX.XXXX)
 DBID(0289)    THE FOLLOWING OBIDS SPECIFIED:    0024
 ALL PAGES     === YOU MAY SPECIFY PAGE(XXXXXXXX) MANY TIMES
 ALL TYPES     === YOU MAY SPECIFY TYPE(XX)
 ALL SUBTYPES  === YOU MAY SPECIFY SUBTYPE(XX)
================================================================================

DSN1212I DSN1LGRD FIRST LOG  RBA ENCOUNTERED 00011DDDFC3D

DSN1224I SPECIFIED LOG RBASTART 00011DDDFC00 COULD NOT BE LOCATED

DSN1213I DSN1LGRD LAST LOG  RBA ENCOUNTERED 00011DDE2EC5

DSN1214I NUMBER OF LOG RECORDS READ 0000000000000091
```

The corresponding DSN1LOGP output from LOGGED and NOT LOGGED.

The top part is from the LOGGED tablespace while the bottom part is from the NOT LOGGED tablespace.

DSN1LOGP did not report any logging for the INSERT in the NOT LOGGED tablespace.

UPDATE LOGGED

- Log content formatted

```
URID: 00011DDBABE7   Member         :
LRSN: C39EDE54D352   Primary Auth-id: RASST02      Plan name      : RCUU1150
Date: 01-19-09       Correlation-id : RASST02      Connection-id  : DB2CALL
Time: 17:40:37.60    URID Status    : Committed    Connection Type:
TSO/Batch
------------------------------------------------------------------------------
-
  Table RASST02.IDUGTB8               Database: IDUG09    Tablespace:
IDUG09S

  Update  RID: 0000000262  Log RBA: 00011DDBAD5D  Log LRSN: C39EDE54D3BA

         *C1
          ----------
   New -> 7777777777
   Old -> 1234567890
```

For the LOGGED tablespace, the TO-and-FROM-byte change is logged as anticipated.

DSN1LOGP LOGGED / NOT LOGGED for UPDATE

```
DSN1212I DSN1LGRD FIRST LOG  RBA ENCOUNTERED 00011DDBAD5D

00011DDBAD5D  TYPE( UNDO  REDO )  URID(00011DDBABE7)
              LRSN(C39EDE54D3BA)  DBID(0121)  OBID(0017)  PAGE(00000002)          17:40:37 09.019
              SUBTYPE(UPDATE IN-PLACE IN A DATA PAGE)  CLR(NO)
              PROCNAME(DSNILREP)

     *LRH* 00580073 06000001 0E800001 1DDBABE7 00011DDB ACEA0626 00011DDB ACEAC39E  *               X
C
           DE54D3BA 0000                                                            *  L
     *LG** 08012100 17000000 02000001 1DDBA34E 2B00                                *           t+
     0000  00200162 00181910 000600F7 F7F7F7F7 F7F7F7F7 F700F1F2 F3F4F5F6 F7F8F9F0  *     7777777777
1234567890
-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
DSN1212I DSN1LGRD FIRST LOG  RBA ENCOUNTERED 00011DDDFC3D

00011DDDFC3D  TYPE( UNDO  REDO )  URID(00011DDDFBAD)
              LRSN(C39F08ED5757)  DBID(0121)  OBID(0017)  PAGE(00000002)           20:51:11 09.019
              SUBTYPE(UPDATE IN-PLACE IN A DATA PAGE)  CLR(NO)
              PROCNAME(DSNILREP)

     *LRH* 00580090 06000001 0E800001 1DDDFBAD 00011DDD FBAD0626 00011DDD FBADC39F  *
C
           08ED5757 0000                                                            *
     *LG** 88012100 17000000 02000001 1DDDF47E 2B00                                *h              4=
```

LOGGED

NOT LOGGED

Unexpected output from DSN1LOGP !!??
Why does the UNDO and REDO information appear in the log ?
We'll go over some more fun in a few slides

Looking at the DSN1LOGP output – an unexpected result !

The UPDATE statement is logged for both LOGGED and NOT LOGGED ??!!

However – this is hard to recreate, and we have only observed this scenario in one particular place. Running REXX EXEC, pure CAF, ASM etc. didn't provoke this situation.

When the UPDATE is done within one specific CA component – then it happens. I'm puzzled :-o

DELETE LOGGED

- Log content formatted

```
URID: 00011DDBB3EC   Member         :
LRSN: C39EDE5BD125   Primary Auth-id: RASST02      Plan name      : RCUU1150
Date: 01-19-09       Correlation-id : RASST02      Connection-id  : DB2CALL
Time: 17:40:44.93    URID Status    : Committed    Connection Type:
TSO/Batch
------------------------------------------------------------------------
-
  Table RASST02.IDUGTB8               Database: IDUG09    Tablespace:
IDUG09S

  Delete  RID: 0000000262  Log RBA: 00011DDBB4EF  Log LRSN: C39EDE5BD126

          C1          C2        C3        C4
          ----------  --------  --------  --------
          7777777777  **NULL**  **NULL**  **NULL**
```

The log report from the delete statement for the LOGGED tablespace illustrates the entire row is logged.

DSN1LOGP LOGGED / NOT LOGGED for DELETE

```
DSN1212I DSN1LGRD FIRST LOG  RBA ENCOUNTERED 00011DDBB4EF

00011DDBB4EF  TYPE( UNDO  REDO )  URID(00011DDBB3EC)
              LRSN(C39EDE5BD126)  DBID(0121)  OBID(0017)  PAGE(00000002)        17:40:44 09.019
              SUBTYPE(DELETE IN A DATA PAGE)  CLR(NO)  PROCNAME(DSNIDILS)

     *LRH* 00670073 06000001 0E800001 1DDBB3EC 00011DDB B47C0626 00011DDB B47CC39E   *             @       @C
           DE5BD126 0000                                                             * $J
     *LG** 08012100 17000000 02000001 1DDBAD5D 3900                                 *              )
     0000  002F2062 00181010 02002700 186200F7 F7F7F7F7 F7F7F7F7 F7FF0000 0000FF00   *              7777777777
     0020  000000FF 00000000 00000000 000000

-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

 ALL URIDS     === YOU MAY SPECIFY URID(XXXXXXXXXXX)
 ALL LUWIDS    === YOU MAY SPECIFY LUWID(NNNNNNNN.LLLLLLLL.XXXXXXXXXXX.XXXX)
 DBID(0289)    THE FOLLOWING OBIDS SPECIFIED:    0024
 ALL PAGES     === YOU MAY SPECIFY PAGE(XXXXXXXX) MANY TIMES
 ALL TYPES     === YOU MAY SPECIFY TYPE(XX)
 ALL SUBTYPES  === YOU MAY SPECIFY SUBTYPE(XX)
================================================================================

DSN1212I DSN1LGRD FIRST LOG  RBA ENCOUNTERED 00011DDDFC3D

DSN1224I SPECIFIED LOG RBASTART 00011DDDFC00 COULD NOT BE LOCATED

DSN1213I DSN1LGRD LAST LOG  RBA ENCOUNTERED 00011DDE2EC5

DSN1214I NUMBER OF LOG RECORDS READ 0000000000000091
```

Just like the INSERT – DSN1LOGP only reports the deleteed row in the log for the LOGGED tablespace. Nothing is found for the singleton row deletion when the tablespace is NOT LOGGED.

DSN1LOGP LOGGED Summary

```
TYPE(PAGE SET CONTROL)
LRSN(C39EDE4A0B02)  DBID(0121)  OBID(0017)
SUBTYPE(PAGE SET OPEN)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
IDUG09  .IDUG09S8
BPID:    0
NO. OF PIECES:    32
PIECE     1) GBP-DEPENDENT: NO
             GBPCACHE: CHANGED
             TESTPAGE: NO
             P-LOCK HELD STATE: NONE
             P-LOCK CACHED STATE: NONE
             P-LOCK UPGRADED: NO
             INSTANCE: I
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
TYPE( UNDO  REDO )  URID(00011DDBA2BE)
LRSN(C39EDE4A0CFB)  DBID(0121) OBID(0017) PAGE(00000002)
SUBTYPE(INSERT IN A DATA PAGE) CLR(NO)PROCNAME(DSNISGRT)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
TYPE(PAGE SET CONTROL)
LRSN(C39EDE4A0DB2)  DBID(0121)  OBID(001A)
SUBTYPE(PAGE SET OPEN)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
IDUG09  .IDUGIX8
BPID:    0
NO. OF PIECES:    32
PIECE     1) GBP-DEPENDENT: NO
             GBPCACHE: CHANGED
             TESTPAGE: NO
             P-LOCK HELD STATE: NONE
             P-LOCK CACHED STATE: NONE
             P-LOCK UPGRADED: NO
             INSTANCE: I
```

```
TYPE( UNDO  REDO ) URID(00011DDBA2BE)
LRSN(C39EDE4A0DB3) DBID(0121)  OBID(001A) PAGE(00000003)
SUBTYPE(TYPE 2 INDEX UPDATE)  CLR(NO)  PROCNAME(DSNKINSL)

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

TYPE( UNDO  REDO )  URID(00011DDBABE7)
LRSN(C39EDE54D352)  DBID(0121)  OBID(001A)
PAGE(00000003)
SUBTYPE(TYPE 2 INDEX UPDATE)  CLR(NO)  PROCNAME(DSNKDLE )

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

TYPE( UNDO  REDO )  URID(00011DDBABE7)
LRSN(C39EDE54D398)  DBID(0121)  OBID(001A)
PAGE(00000003)
SUBTYPE(TYPE 2 INDEX UPDATE)  CLR(NO)  PROCNAME(DSNKINSL)

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
TYPE( UNDO  REDO )  URID(00011DDBABE7)
LRSN(C39EDE54D3BA)  DBID(0121)  OBID(0017)
PAGE(00000002)
SUBTYPE(UPDATE IN-PLACE IN A DATA PAGE)  CLR(NO)
PROCNAME(DSNILREP)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
TYPE( UNDO  REDO )  URID(00011DDBB3EC)
LRSN(C39EDE5BD125)  DBID(0121)  OBID(001A)
PAGE(00000003)
SUBTYPE(TYPE 2 INDEX UPDATE)  CLR(NO)  PROCNAME(DSNKDLE )

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
TYPE( UNDO  REDO )  URID(00011DDBB3EC)
LRSN(C39EDE5BD126)  DBID(0121)  OBID(0017)
PAGE(00000002)
```

This is the entire DSN1LOGP report for the previous operations for the LOGGED tablespace – everything as expected.

DSN1LOGP NOT LOGGED Summary

```
TYPE(PAGE SET CONTROL)
LRSN(C39F08DF87EE)  DBID(0121)  OBID(0017)
SUBTYPE(PAGE SET OPEN)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
IDUG09  .IDUG09S8
BPID:    0
NO. OF PIECES:    32
NOT LOGGED
PIECE     1) GBP-DEPENDENT: NO
             GBPCACHE: CHANGED
             NON-RECOVERABLE
             TESTPAGE: NO
             P-LOCK HELD STATE: NONE
             P-LOCK CACHED STATE: NONE
             P-LOCK UPGRADED: NO
             INSTANCE: I
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
TYPE(PAGE SET CONTROL)
LRSN(C39F08DFBC23)  DBID(0121)  OBID(001A)
SUBTYPE(PAGE SET OPEN)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
IDUG09  .IDUGIX8
BPID:    0
NO. OF PIECES:    32
NOT LOGGED
PIECE     1) GBP-DEPENDENT: NO
             GBPCACHE: CHANGED
             NON-RECOVERABLE
             TESTPAGE: NO
             P-LOCK HELD STATE: NONE
             P-LOCK CACHED STATE: NONE
             P-LOCK UPGRADED: NO
             INSTANCE: I
```

```
TYPE( UNDO  REDO )  URID(00011DDDFBAD)
LRSN(C39F08ED5757)  DBID(0121)  OBID(0017)
PAGE(00000002)
SUBTYPE(UPDATE IN-PLACE IN A DATA PAGE)  CLR(NO)
PROCNAME(DSNILREP)

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

TYPE(PAGE SET CONTROL)
LRSN(C39F0949698F)  DBID(0121)  OBID(0017)
SUBTYPE(PAGE SET CLOSE)
```

NOT LOGGED definitely does less logging.

Nothing could be found in the log from INSERT and DELETE processing – only UPDATE processing.

The corresponding DSN1LOGP report for the NOT LOGGED tablespace.

There is no doubt that NOT LOGGED tablespaces does cause less logging. And the DSN1LOGP report is a lot small if the UPDATE statement is executed outside CA RC/Update – I still haven't found the cause

DSN1LOGP procedure / scenario summary – let's look at the NOT logged issue where logging occurs – sometimes

1) QUIESCE tablespace : x'0178BDE832FF'
2) Update date column to '2010-03-17'
3) QUIESCE tablespace : x'0178BDEB18C1'
4) ARCHIVE LOG MODE(QUIESCE)
5) ALTER TABLESPACE NOT LOGGED
6) QUIESCE tablespace : x'0178BDF143D0'
7) Update date column to '2222-10-10'
8) Tablespace is in RW,ICOPY
9) QUIESCE tablespace : x'0178BDF3C563'
10) ARCHIVE LOG MODE(QUIESCE)
11) Time to look at DSN1LOGP

Lets' look back at the "strange issue" where an UPDATE was logged even tough the tablespace was in a NOT LOGGED status.

This is the scenario illustrating what exactly is happening on the log (DSN1LOGP details can be provided).

DSN1LOGP procedure / scenario summary

- DSN1221E DSNJSLR ERROR RETCODE=0000000C REASON CODE= 00D1002A VSAM RETURN CODE=0008 ERROR CODE=00A8
- Since the log-range is within an ARCHIVE log – why doesn't it go after that one. I could specify the specific archive log dataset . . . . . (next slide)

```
LALOGI R14     --------- Log Analyzer Log Inventory List -------- 10/03/17 13:11
COMMAND ===>                                                  SCROLL ===> CSR

Log Type    ==> * ( A - Active, R - Archive, * - All )      Select Members ==> N
Log Copy    ==> * ( 1 - Copy 1, 2 - Copy 2 , * - All )
BSDS Name   ==> D91A.BSDS01
------------------------------------------------------------------------ RASST02
Begin Date ==>                      End Date ==>
Begin RBA  ==> 0178BDE832FF         End RBA  ==> 0178BDF3C563
Begin LRSN ==>                      End LRSN ==>

Log      Log        +-------- Log Dataset Begin / Log Dataset End -----------+
Type     Copy       UTC Date       UTC Time       Log RBA          Log LRSN
ACTIVE    1         2010/03/17     13:37:37       0178B370B000     N/A
                    2010/03/17     16:36:17       0178BDF93FFF     N/A
DSN     : D91A.LOGCOPY1.DS01
CREATED: 2007/08/02 10:41:00   STATUS: REUSABLE,TRUNCATED
-------------------------------------------------------------------------------
ARCHIVE   1         2010/03/17     13:37:37       0178B370B000     N/A
                    2010/03/17     16:36:17       0178BDF93FFF     N/A
DSN     : D91A.ARCHLOG1.A0009397
CREATED: 2010/03/17 12:36:00   UNIT: SYSDA      VOLSER: LOG004     CATALOGED
-------------------------------------------------------------------------------
ACTIVE    2         2010/03/17     13:37:37       0178B370B000     N/A
                    2010/03/17     16:36:17       0178BDF93FFF     N/A
DSN     : D91A.LOGCOPY2.DS02
CREATED: 2008/02/28 11:02:00   STATUS: REUSABLE,TRUNCATED
-------------------------------------------------------------------------------
```

Even though you have arcived the LOG using

-ARCHIVE LOG MODE(QUIESCE)

You might not be able to use DSN1LOGP without specifying the ARCHIVE LOG specifically. DSN1LOGP won't automatically pick THE archive log if the same log-range is to be found on one of the ACTIVE logs (please see next page)

2010 IDUG North America

DSN1LOGP procedure / scenario summary

- Specifying the specific archive log-dataset – DSN1LOGP succeeds

```
000001 //RASST02G JOB  ('106720000'),'SDR-CA-DK',CLASS=A,MSGCLASS=X,
000002 //     NOTIFY=RASST02,SCHENV=D91A,
000003 //  USER=RASST02,PASSWORD=
000004 //STEP1     EXEC PGM=DSN1LOGP,PARM='PRINT'
000005 //STEPLIB   DD DSN=DB2.DB2910.SDSNLOAD,DISP=SHR
000006 //SYSPRINT DD SYSOUT=*
000007 //SYSPRINT DD SYSOUT=*
000008 //*BSDS      DD DSN=D91A.BSDS01,DISP=SHR
000009 //ARCHIVE   DD DSN=D91A.ARCHLOG1.A0009397,DISP=SHR
000010 //SYSIN     DD *              GENERATED STATEMENT
000011   RBASTART(0178BDEB18C2) RBAEND(0178BDF3C563) SUMMARY(NO) DBID(4C82)
000012 /*
****** ***************************** Bottom of Data ***************************
```

- Lets look at the UPDATE statement for the tablespace when it was LOGGED and NOT LOGGED: IDUG DSN1LOGP UPDATE.txt
- Let's see if there's a difference between UPDATE and INSERT for this specific scenario. IDUG DSN1LOGP INSERT.txt

Once the ARCHIVE LOG is specified in the JCL – then we get the output.

Lets look into the DSN1LOGP – weird how UPDATE is handled.

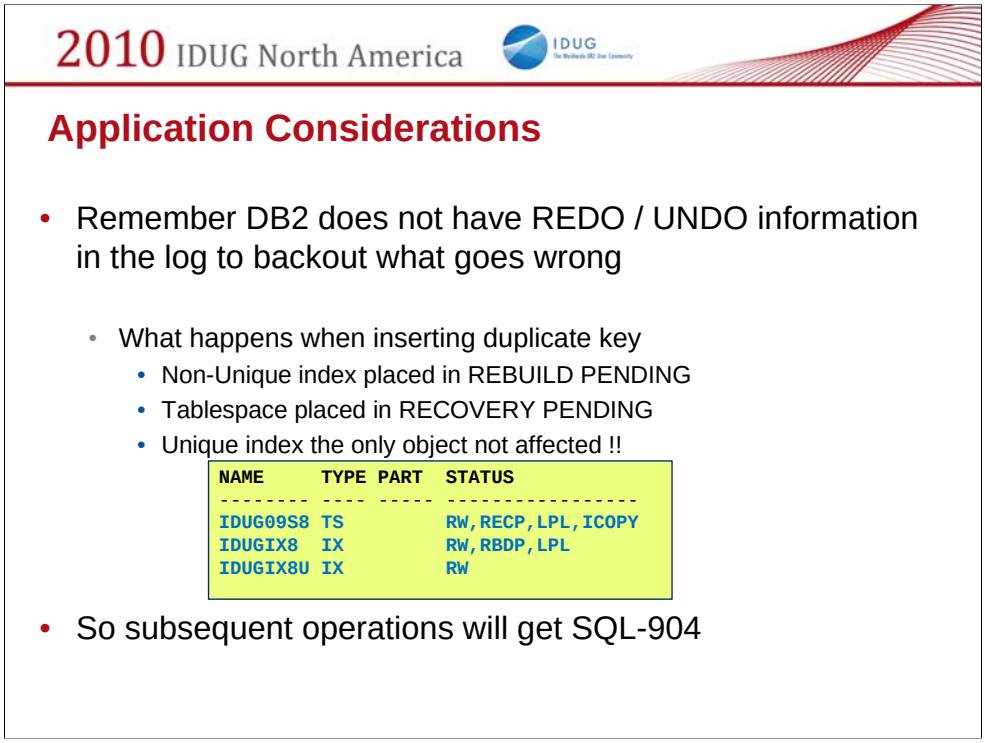
Even though the tablespace is NOT LOGGED, DB2 creates UNDO-REDO for one specific scenario – SOMETIMES.

Application Considerations

Make sure you can climb out of the hole you're digging !!!!!!

Before you get too exited using NOT LOGGED tablespaces – be sure you know all the application implications. You might want to reconsider and think about WHEN you find this feature useful.

Let's have a closer look at what CAN happen . . . . . .



Earlier we discussed that UNDO/REDO information isn't registered in the log for NOT LOGGED tablespaces – which means – if something goes "south" in an application program, DB2 has no way to backout what happened.

One example – a table resides in a NOT LOGGED tablespace. The table in this tablespace has a UNIQUE key/index.

When the tablespace is NOT LOGGED and an application attempts to insert a duplicate row :

1) SQL-803 is issued.
2) Any NON-UNIQUE index is placed in REBUILD PENDING
3) The tablespace is placed in RECOVER PENDING
4) Both the tablespace and the unique index is placed in the LPL list – and the LPL status will not go away by doing the –STA command

At this point – every SQL statement against this table will receive an UNAVAILABLE RESOURCE message. You are in a recovery situation.

Application Considerations

- If multiple applications / jobs will be manipulating the same NOT LOGGED tablespace
  - Be careful to run in parallel
  - Take an image copy once one application is done to avoid having to remove updates from first application if second application fails
- First update operation after tablespace altered to NOT LOGGED will place tablespace in ICOPY

Looking at the previous slide and the result of a NEGATIVE SQLCODE – you might consider running parallel applications when one tablespace is NOT LOGGED.

Also – make sure to take a FULL Image Copy the very moment the tablespace is altered to be NOT LOGGED in order to ease a potential recovery.

Backup and Recovery Considerations

Recovery Considerations

- Where can we recover to – scenario outline is:
  - Create LOGGED tablespace
  - Execute Image Copy 1
  - Insert one row
  - Execute Image Copy 2
  - Alter tablespace NOT LOGGED (status RW)
  - Insert one more row (ICOPY status for the tablespace)
  - Execute Image Copy 3 (status now RW)
  - Insert another row (status ICOPY again)
  - Execute QUIESCE – which ended successfully
- Time to test some recovery scenarios

Using NOT LOGGED tablespace might place you in recovery scenarios you're not used to, so let's have a closer look at which point in time a recovery is possible dealing with NOT LOGGED tablespaces.

This scenario describes different "normal recovery points of consistence" by using QUIESCE and altering the tablespace from LOGGED to NOT LOGGED.

The following slides will illustrate which QUIESCE points we can recover to when dealing with tablespaces being altered to be NOT LOGGED.

Recovery Considerations

- Recovery to the QUIESCE point taken after tablespace was altered to be NOT LOGGED

```
DSNUGUTC -  RECOVER TABLESPACE(IDUG09.IDUG09S8 DSNUM ALL) TORBA X'00011DF05E6D'

DSNUCASA -  RECOVERY OF NOT LOGGED TABLESPACE IDUG09.IDUG09S8   CANNOT PROCEED
                BECAUSE THE TOLOGPOINT OR TORBA SPECIFIED IS NOT A RECOVERABLE POINT

DSNUGSRX - TABLESPACE IDUG09.IDUG09S8 IS IN INFORMATIONAL COPY PENDING STATUS
DSNUCBDR - RECOVERY COMPLETE, ELAPSED TIME=00:00:00
DSNUGBAC - UTILITY EXECUTION TERMINATED, HIGHEST RETURN CODE=8
```

- To DB2 – this is a “Non Recoverable Point”
  - “No” log records exist for the operations

The first recovery attempts to recover to the QUIESCE point taken immediately after the tablespace was altered to NOT LOGGED.

To DB2 this is not a valid recovery point. DB2 will look at SYSCOPY to see when the tablespace was altered to be NOT LOGGED and then rejects to execute the recovery since log-records are not written for DML statements.

Recovery considerations

- Recover to an Image Copy taken during the period where the tablespace was NOT LOGGED.

```
DSNUGUTC -  RECOVER TABLESPACE(IDUG09.IDUG09S8 DSNUM ALL) TOCOPY
                  RASST02.IDUG09.IDUG09S8.IC003DSNUCBAL –
                  THE IMAGE COPY DATA SET RASST02.IDUG09.IDUG09S8.IC003 WITH
                                          DATE=DATE=20090120 AND TIME=180143
IS PARTICIPATING IN RECOVERY OF TABLESPACE IDUG09.IDUG09S8
DSNUCBMD - MERGE STATISTICS FOR TABLESPACE IDUG09.IDUG09S8   -
        NUMBER OF COPIES=1
        NUMBER OF PAGES MERGED=3
        ELAPSED TIME=00:00:02
DSNUCBDR - RECOVERY COMPLETE, ELAPSED TIME=00:00:03
DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

- This is possible since the image copy is a consistent point

This scenario describes a recovery to an image copy taken while the tablespace was defined as NOT LOGGED.

Unlike the RBA recovery in the previous scenario, this is valid since the IMAGE COPY is a consistent point.

Recovery considerations

- Recover to a point in time PRIOR to the tablespace being altered to be NOT LOGGED

```
DSNUGUTC - RECOVER TABLESPACE(IDUG09.IDUG09S8 DSNUM ALL) TOCOPY
                RASST02.IDUG09.IDUG09S8.IC001
DSNUCBAL - THE IMAGE COPY DATA SET RASST02.IDUG09.IDUG09S8.IC001
                WITH DATE=200900120 AND TIME=175842
                IS PARTICIPATING IN RECOVERY OF TABLESPACE IDUG09.IDUG09S8
DSNUCBMD - MERGE STATISTICS FOR TABLESPACE IDUG09.IDUG09S8   -
                        NUMBER OF COPIES=1
                        NUMBER OF PAGES MERGED=3
                        ELAPSED TIME=00:00:03
DSNUCBDR - RECOVERY COMPLETE, ELAPSED TIME=00:00:03
DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

- Also possible since this is a point of consistency

The last scenario attempts to recover to a point in time where the tablespace was defined as LOGGED.

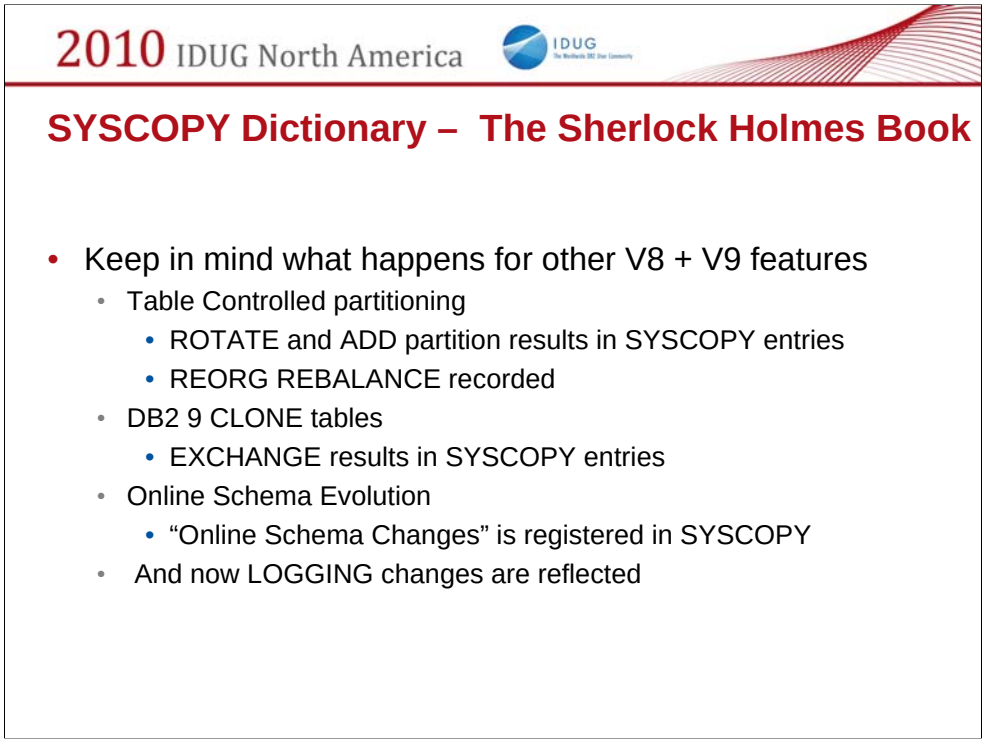
This is also a valid recovery point.

Recovery considerations

- A FULL Recover will work as long as an image copy really does exist
- Will recover to the most recent recoverable point
- No need to worry about the previous scenarios – or ??

The last scenario attempts to recover to a point in time where the tablespace was defined as LOGGED.

This is also a valid recovery point.



If we look back at what has happened in DB2 V8 and DB2 9 when exploiting Online Schema Changes – DB2 stores a lot of these dynamic object changes in SYSCOPY.

At first sight this might look weird, but considering the recovery issues dealing with adding / rotating partitions, changing the limitkeys, turning off logging etc., it is very convenient to have a "one-stop-shop" for deciding if an object can be recovered.

So SYSCOPY is a dictionary or video recording of actions done for objects in order to determine recoverability.

SYSCOPY Dictionary

- A few new column values for ICTYPE and STYPE can be found in SYSCOPY.

ICTYPE		STYPE	ACTION
C		L	CREATE tablespace LOGGED
C		O	CREATE tablespace NOT LOGGED
A		L	ALTER tablespace LOGGED
A		O	ALTER tablespace NOT LOGGED

Just to recap – ICTYPE and STYPE columns in SYSCOPY has some additional values available in DB2 9.

One major new attribute is that the creation of tablespaces are stored in SYSCOPY – not only when the logging attribute changes.

Sources used

- Primarily a live DB2 9 NFM system
- SQL Reference Guide
- “The DB2 Bible” – Database Administration Guide

The statement IBM has been stressing since NOT LOGGED tablespaces was introduced is definitely true – don’t use NOT LOGGED to save application CPU.

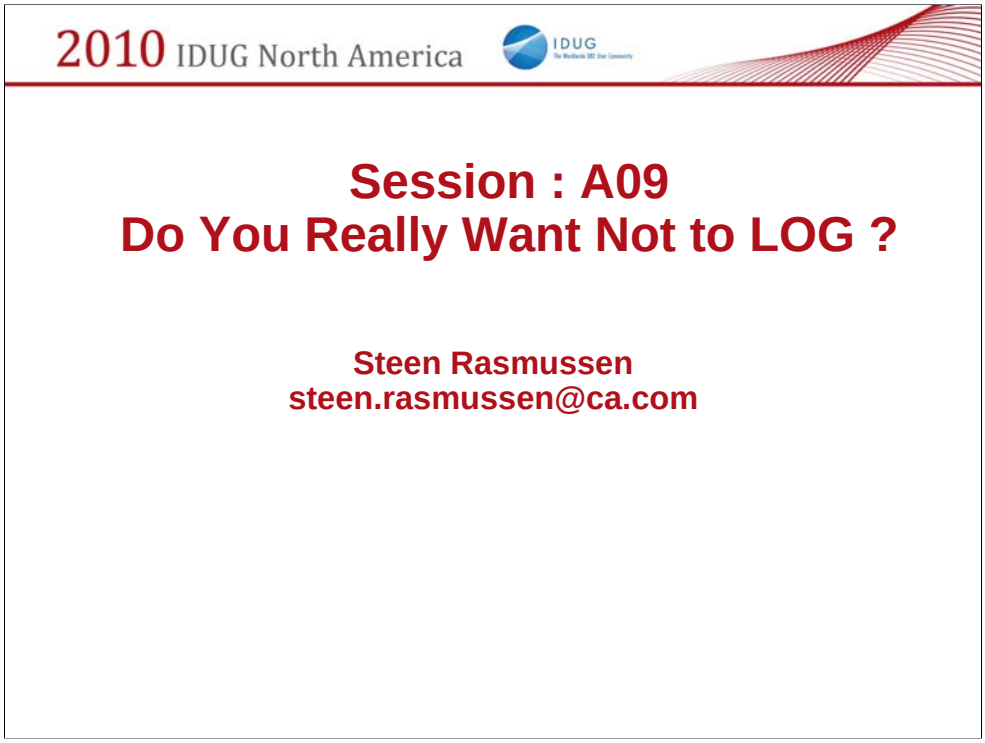
If using NOT LOGGED – make sure the application environment is analyzed and create a recovery plan WHEN something goes wrong.

Conclusion

- Don't use NOT LOGGED to save CPU for SQL operations
- MQT's seem to be an obvious choice
  - Restart half way through REFRESH not possible
- I've heard another customer considering a HEAVY INSERT batch application (like staging table)
- IBM LOAD RESUME YES / REPLACE
- If NOT LOGGED used
  - Make sure you always are recoverable
  - Consider if parallel application processing is a good idea

The statement IBM has been stressing since NOT LOGGED tablespaces was introduced is definitely true – don't use NOT LOGGED to save application CPU.

If using NOT LOGGED – make sure the application environment is analyzed and create a recovery plan WHEN something goes wrong.



Steen Rasmussen is a Sr. Services Engineering Architect currently instrumental in the ongoing development and support of the CA DB2 tools. In 1985 Steen started as an IMS/DB2 DBA at a major insurance company in Denmark working with all aspects of DB2 - like tuning, application design and implementation, education of developers, backup and recovery planning and automation of housekeeping processes. In 1995 Steen became a technical manager at PLATINUM Technology managing technical support and presales for the DB2 products.

Steen has been working with DB2 for more than 20 years starting with DB2 Release 1.0. Besides from providing support for the CA teams in the field as well as internal groups working with DB2, Steen also is a frequent speaker at IDUG in North America and Europe as well as local DB2 User Groups around the world.