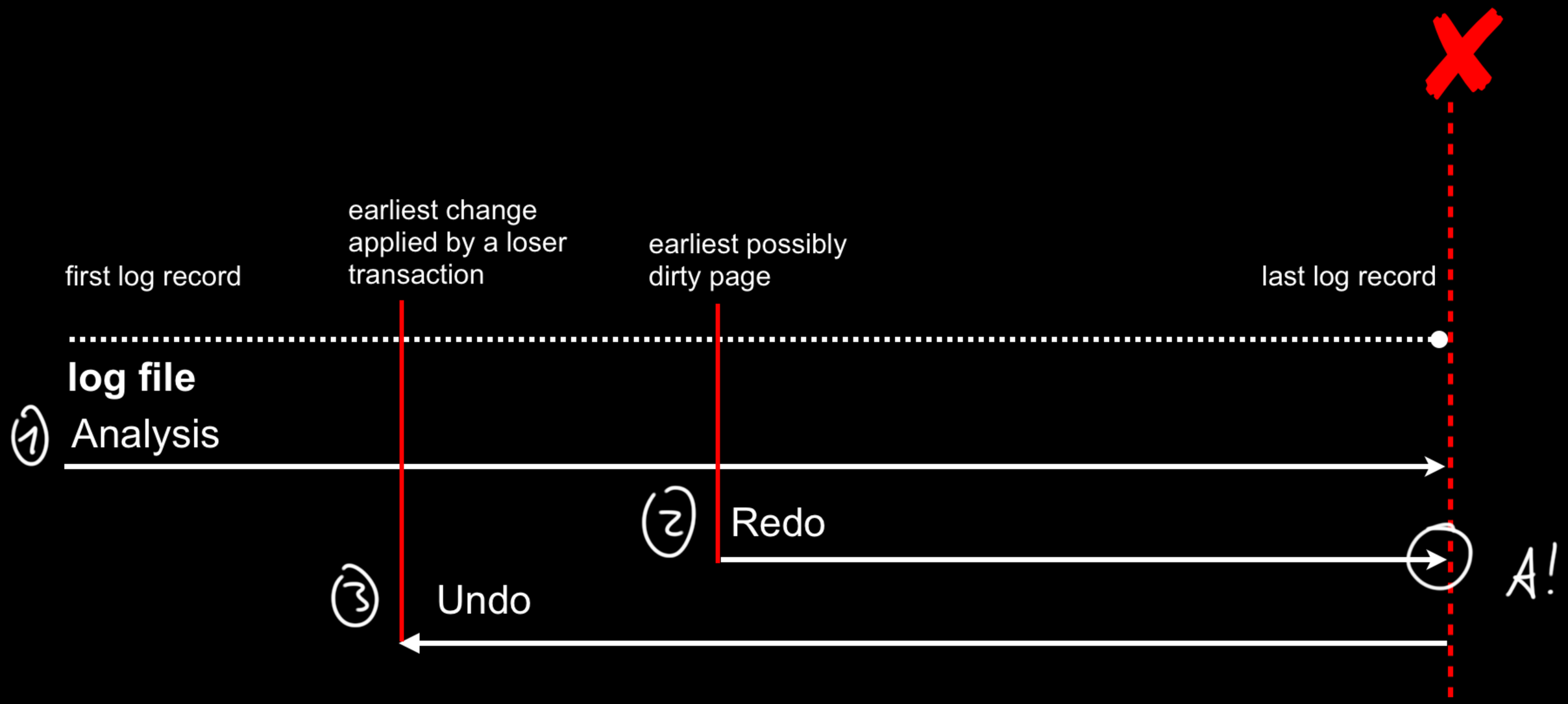
# 3 Phases of ARIES

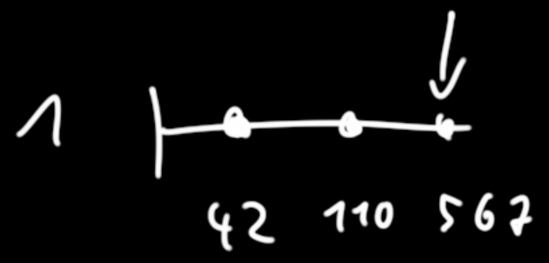





notice: for the moment no checkpointing!

# Transaction Table TT

| TT | |
|---|---|
| TaID | lastLSN |
| 1 | 567 |
| 2 | 42 |
| 7 | 67 |
| 3 | 12 |




TaID: key of the transaction

lastLSN: LSN of the most recent log record seen for this transaction, i.e. the **latest** change done by this transaction

# Dirty Page Table DPT

| DPT | |
|---|---|
| pageID | recoveryLSN |
| 42 | 567 |
| 46 | 568 |
| 77 | 34 |
| 3 | 42 |

# Transaction Table TT

| TT | |
|---|---|
| TaID | lastLSN |
| 1 | 567 |
| 2 | 42 |
| 7 | 67 |
| 3 | 12 |

TaID: key of the transaction

lastLSN: LSN of the most recent log record seen for this transaction, i.e. the **latest** change done by this transaction

**assumptions**: page may contain changes from losers and winners, i.e. two concurrent transcations may operate **on the same page**

# Dirty Page Table DPT

| DPT | |
|---|---|
| pageID | recoveryLSN |
| 42 | 567 |
| 46 | 568 |
| 77 | 34 |
| 3 | 42 |

pageID: key of a page

recoveryLSN: LSN of **first** log record that made this page dirty, i.e. the **earliest** change done to this page

p42

T1 → commits

T2 → aborts

① → tuple-wise locking

NO-buffer:

② STEAL

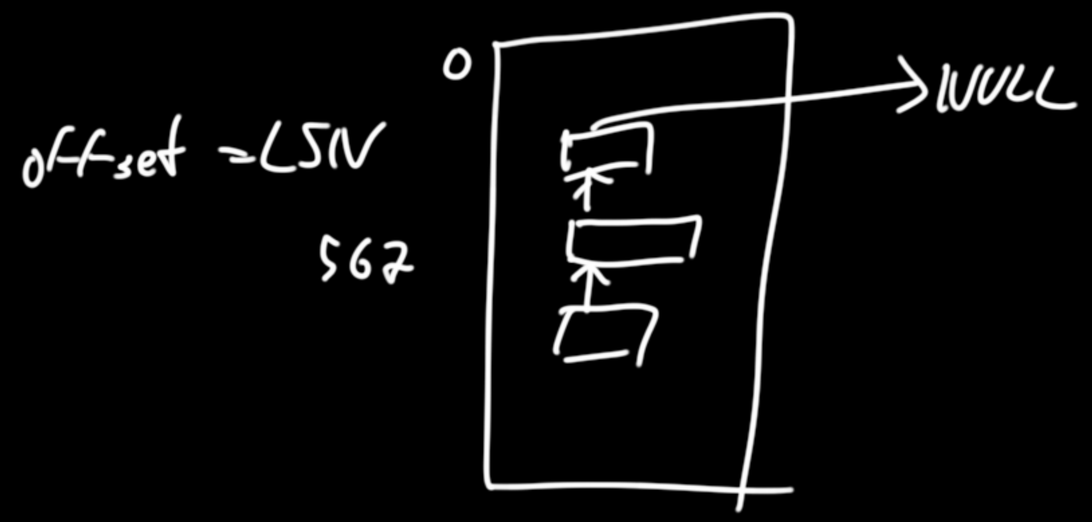
③ NO-FORCE

# The Log Records in Detail

All Log Records:

[prevLSN, TaID, type]

420 | 42 |




prevLSN: (Previous Log Sequence Number): link to the previous LSN for this TA (if not first entry)

TaID: foreign key to unique transaction ID

# The Log Records in Detail

All Log Records:

[prevLSN, TaID, type]

prevLSN: (Previous Log Sequence Number): link to the previous LSN for this TA (if not first entry)

TaID: foreign key to unique transaction ID

type: kind of log record; either: update, commit or compensation (for undoing updates)

# The Log Records in Detail: Update Log Records

All Log Records:

[prevLSN, TaID, type]

Update Log Record:

[prevLSN, TaID, “update“, pageID, redo info, undo info]

!WAL!

# The Log Records in Detail: Update Log Records

All Log Records:

[prevLSN, TaID, type]

Update Log Record:

[prevLSN, TaID, “update“, pageID, redo info, undo info]

physical or physiological

pageID: foreign key to pageID

# The Log Records in Detail: Update Log Records

All Log Records:

[prevLSN, TaID, type]

Update Log Record:

[prevLSN, TaID, “update“, pageID, redo info, undo info]

pageID: foreign key to pageID

redo info: information how to redo the change reflected by this log record (physiological)

# The Log Records in Detail: Update Log Records

All Log Records:

[prevLSN, TaID, type]

logical undo : concurrency in index structure

Update Log Record:

[prevLSN, TaID, "update", pageID, redo info, undo info]

For the moment

ARIES : redo : physiological
undo : (logical) physiological

pageID: foreign key to pageID

redo info: information how to redo the change reflected by this log record (physiological)

undo info: information how to undo the change reflected by this log record (logical)

# Example Log File

LSN ~ version

LSN: [prevLSN, TaID, type] all

LSN: [prevLSN, TaID, “update“, pageID, redo info, undo info] “update“

DB buffer

page 42

LSN=-

a=77

b=55

page 46

LSN=-

c=22

log

| TT | |
|---|---|
| TaID | lastLSN |

| DPT | |
|---|---|
| pageID | recoveryLSN |

# Example Log File

LSN: [prevLSN, TaID, type] — all

LSN: [prevLSN, TaID, “update“, pageID, redo info, undo info] — “update“

DB buffer

page 42

LSN=1

a=78

b=55

page 46

LSN=-

c=22

log

1: [-, 1, “update“, 42, a+=1, a-=1]

| TT | |
|---|---|
| TaID | lastLSN |
| 1 | 1 |

| DPT | |
|---|---|
| pageID | recoveryLSN |
| 42 | 1 |

# Example Log File

LSN: [prevLSN, TaID, type] — all

LSN: [prevLSN, TaID, “update“, pageID, redo info, undo info] — “update“

DB buffer

page 42

LSN=2

a=78

b=58

page 46

LSN=-

c=22

log

1: [-, 1, “update“, 42, a+=1, a-=1]

2: [-, 2, “update“, 42, b+=3, b-=3]

**TT**

| TaID | lastLSN |
|---|---|
| 1 | 1 |
| 2 | 2 |

**DPT**

| pageID | recoveryLSN |
|---|---|
| 42 | 1 |

# Example Log File

LSN: [prevLSN, TaID, type] — all

LSN: [prevLSN, TaID, “update“, pageID, redo info, undo info] — “update“

DB buffer

page 42

LSN=2

a=78

b=58

page 46

LSN=3

c=24

log

1: [-, 1, “update“, 42, a+=1, a-=1]
2: [-, 2, “update“, 42, b+=3, b-=3]
3: [2, 2, “update“, 46, c+=2, c-=2]

**TT**

| TaID | lastLSN |
|---|---|
| 1 | 1 |
| 2 | 3 |

**DPT**

| pageID | recoveryLSN |
|---|---|
| 42 | 1 |
| 46 | 3 |

# Example Log File

LSN: [prevLSN, TaID, type] all

LSN: [prevLSN, TaID, “update“, pageID, redo info, undo info] “update“

DB buffer

page 42

LSN=4

a=78

b=59

page 46

LSN=3

c=24

log

1: [-, 1, “update“, 42, a+=1, a-=1]
2: [-, 2, “update“, 42, b+=3, b-=3]
3: [2, 2, “update“, 46, c+=2, c-=2]
4: [1, 1, “update“, 42, b+=1, b-=1]

**TT**

| TaID | lastLSN |
|---|---|
| 1 | 4 |
| 2 | 3 |

**DPT**

| pageID | recoveryLSN |
|---|---|
| 42 | 1 |
| 46 | 3 |

# Example Log File

LSN: [prevLSN, TaID, type] — all

LSN: [prevLSN, TaID, “update“, pageID, redo info, undo info] — “update“

DB buffer

page 42

LSN=4

a=78

b=59

page 46

LSN=3

c=24

log

1: [-, 1, “update“, 42, a+=1, a-=1]
2: [-, 2, “update“, 42, b+=3, b-=3]
3: [2, 2, “update“, 46, c+=2, c-=2]
4: [1, 1, “update“, 42, b+=1, b-=1]
5: [3, 2, “commit“]

**TT**

| TaID | lastLSN |
|---|---|
| 1 | 4 |

**DPT**

| pageID | recoveryLSN |
|---|---|
| 42 | 1 |
| 46 | 3 |

# The Log Records in Detail: Compensation Log Records

1. Analysis
2. Redo → handle CLRs
3. Undo for every undo operation: write CLR

All Log Records:

[prevLSN, TaID, type]

Update Log Record:

[prevLSN, TaID, “update“, pageID, redo info, undo info]

Compensation Log Record (CLR, for undoing updates):

[prevLSN, TaID, “compensation“, **redoTheUndo** info, undoNextLSN]

redoTheUndo info: information how to redo the undo (reflected by the log record creating this CLR)

# The Log Records in Detail: Compensation Log Records

In general, and as mentioned above, in the multi-level ARIES method, CLRs may be (fully) logical not showing a pageID.

However, I simplified the explanation of ARIES in this video such that every log record is either physical or physiological, but NOT logical.

Therefore, the schema of the CLRs show a pageID (as it does in the following example).

All Log Records:

[prevLSN, TaID, type]

Update Log Record:

[prevLSN, TaID, “update“, pageID, redo info, undo info]

Compensation Log Record (CLR, for undoing updates):

[prevLSN, TaID, “compensation“, pageID, **redoTheUndo** info, undoNextLSN]

redoTheUndo info: information how to redo the undo (reflected by the log record creating this CLR)

undoNextLSN: link to the next log record to be undone for this transaction

i.e., if a log record LR wrote this CLR, we set CLR.undoNextLSN := LR.prevLSN

# Example Log File

| | |
|---|---|
| LSN: [prevLSN, TaID, type] | all |
| LSN: [prevLSN, TaID, “update“, pageID, redo info, undo info] | “update“ |
| LSN: [prevLSN, TaID, “compensation“, redoTheUndo info, undoNextLSN] | “compensation“ |

DB buffer

page 42

LSN=4

a=78

b=59

page 46

LSN=3

c=24

log

1: [-, 1, “update“, 42, a+=1, a-=1]
2: [-, 2, “update“, 42, b+=3, b-=3]
3: [2, 2, “update“, 46, c+=2, c-=2]
4: [1, 1, “update“, 42, b+=1, b-=1]
5: [3, 2, “commit“]

① Analysis ② Redo

**TT**

| TaID | lastLSN |
|---|---|
| 1 | 4 |

**DPT**

| pageID | recoveryLSN |
|---|---|
| 42 | 1 |
| 46 | 3 |

min min(dirty page LSN)

# Example Log File

| | |
|---|---|
| LSN: [prevLSN, TaID, type] | all |
| LSN: [prevLSN, TaID, “update“, pageID, redo info, undo info] | “update“ |
| LSN: [prevLSN, TaID, “compensation“, redoTheUndo info, undoNextLSN] | “compensation“ |

DB buffer

page 42

LSN=6

a=78

b=58

page 46

LSN=3

c=24

log

1: [-, 1, “update“, 42, a+=1, a-=1]
2: [-, 2, “update“, 42, b+=3, b-=3]
3: [2, 2, “update“, 46, c+=2, c-=2]
4: [1, 1, “update“, 42, b+=1, b-=1]
5: [3, 2, “commit“]
6: [4, 1, “compensation“, 42, b-=1, 1]

**TT**

| TaID | lastLSN |
|---|---|
| 1 | 6 |

**DPT**

| pageID | recoveryLSN |
|---|---|
| 42 | 1 |
| 46 | 3 |

# Example Log File

| | |
|---|---|
| LSN: [prevLSN, TaID, type] | all |
| LSN: [prevLSN, TaID, “update“, pageID, redo info, undo info] | “update“ |
| LSN: [prevLSN, TaID, “compensation“, redoTheUndo info, undoNextLSN] | “compensation“ |

DB buffer

page 42

LSN=7

a=77

b=58

page 46

LSN=3

c=24

log

(1) Analysis (2) Redo

1: [-, 1, “update“, 42, a+=1, a-=1]
2: [-, 2, “update“, 42, b+=3, b-=3]
3: [2, 2, “update“, 46, c+=2, c-=2]
4: [1, 1, “update“, 42, b+=1, b-=1]
5: [3, 2, “commit“]
6: [4, 1, “compensation“, 42, b-=1, 1]
7: [6, 1, “compensation“, 42, a-=1, - ]

**TT**

| TaID | lastLSN |
|---|---|
| 1 | 7 |

**DPT**

| pageID | recoveryLSN |
|---|---|
| 42 | 1 |
| 46 | 3 |

# Example Log File

| | |
|---|---|
| LSN: [prevLSN, TaID, type] | all |
| LSN: [prevLSN, TaID, “update“, pageID, redo info, undo info] | “update“ |
| LSN: [prevLSN, TaID, “compensation“, redoTheUndo info, undoNextLSN] | “compensation“ |

DB buffer

page 42

LSN=7

a=77

b=58

page 46

LSN=3

c=24

log

1: [-, 1, “update“, 42, a+=1, a-=1]
2: [-, 2, “update“, 42, b+=3, b-=3]
3: [2, 2, “update“, 46, c+=2, c-=2]
4: [1, 1, “update“, 42, b+=1, b-=1]
5: [3, 2, “commit“]
6: [4, 1, “compensation“, 42, b-=1, 1]
7: [6, 1, “compensation“, 42, a-=1, - ]

**TT**

| TaID | lastLSN |
|---|---|

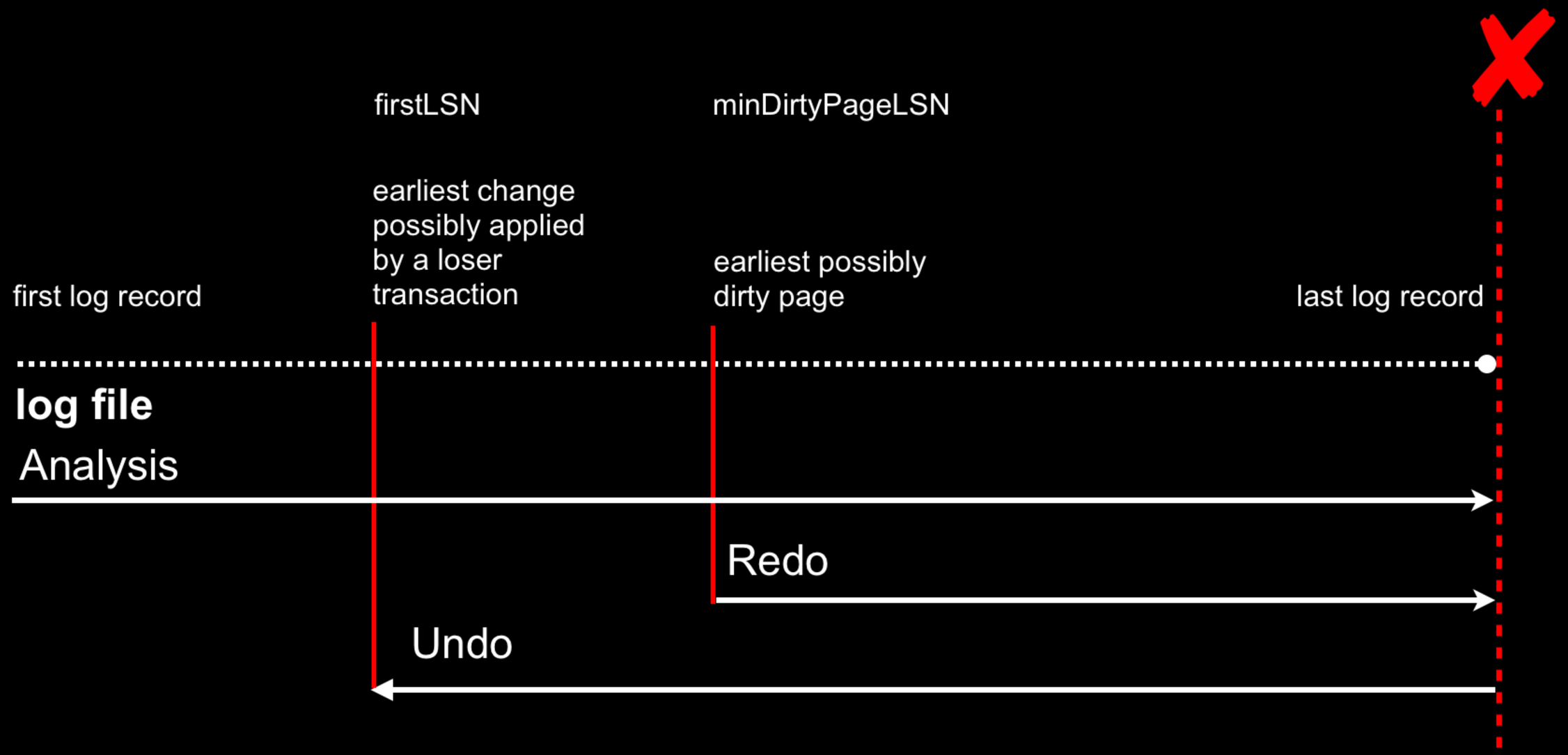
f: ulLSN

**DPT**

| pageID | recoveryLSN |
|---|---|
| 42 | 1 |
| 46 | 3 |

# Starting Points

minDirtyPageLSN := SELECT min(recoveryLSN) FROM DPT;

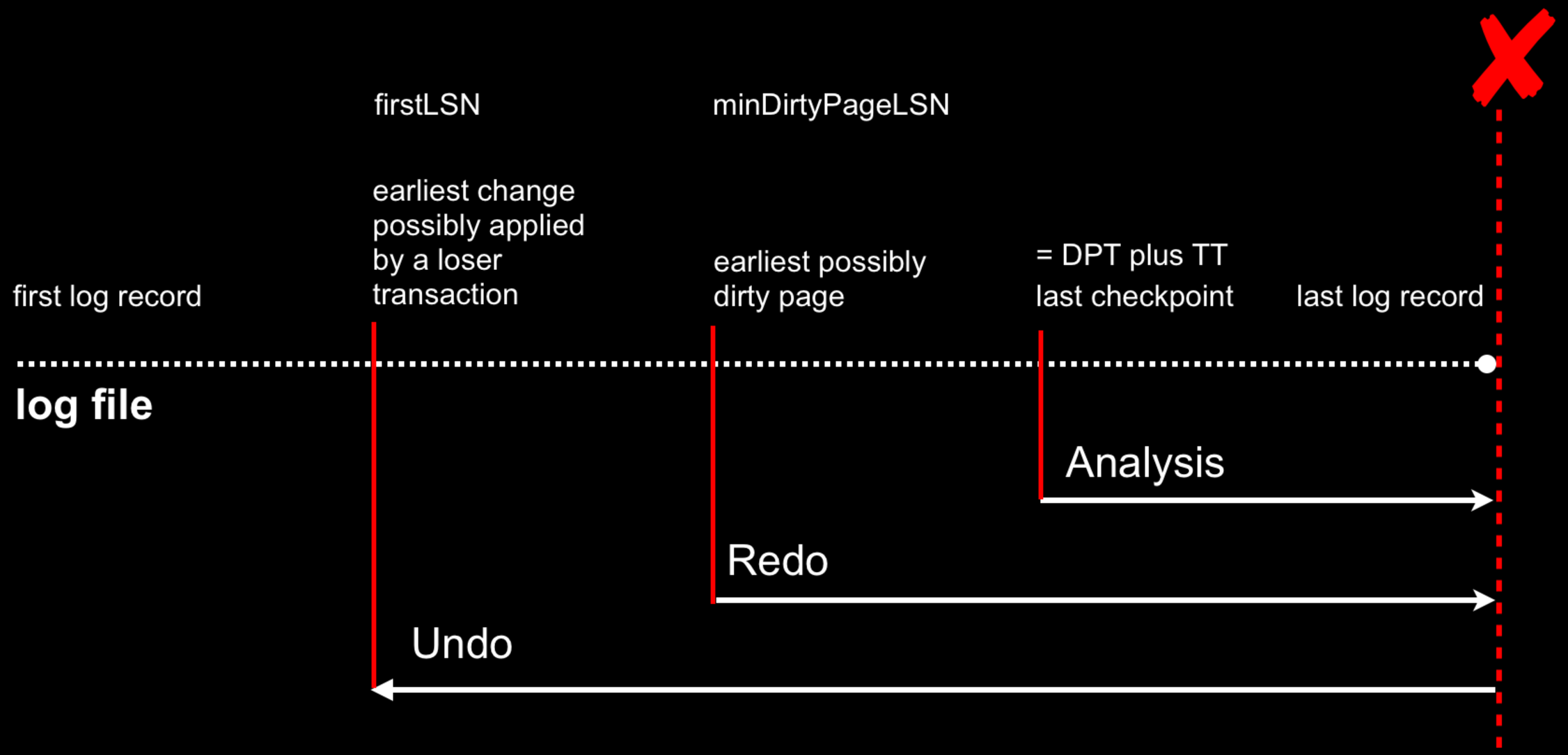



notice: for the moment no checkpointing!

note: relative starting points of the phases may differ

# Fuzzy Checkpoint

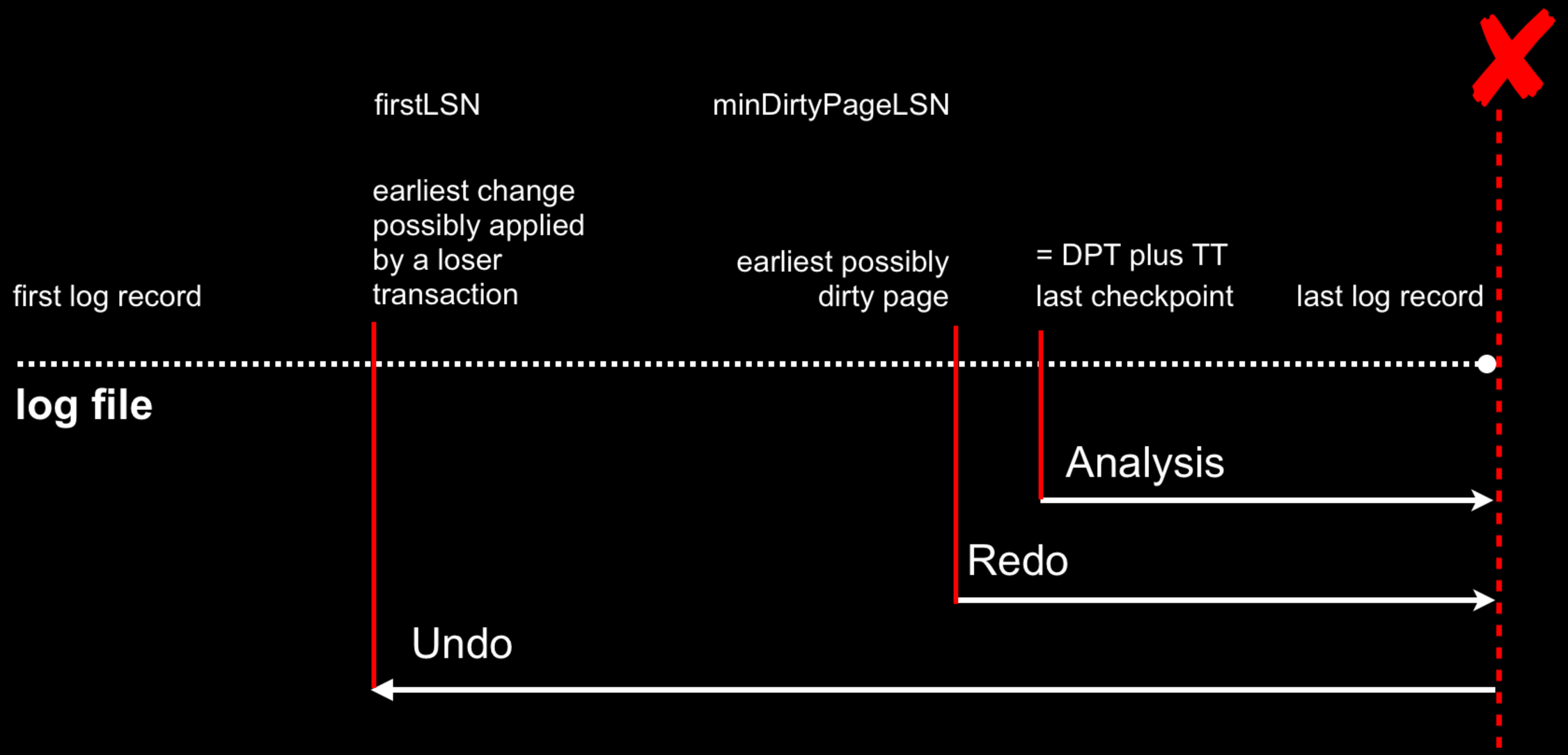
minDirtyPageLSN := SELECT min(recoveryLSN) FROM DPT;




note: relative starting points of the phases may differ

# Background Write Thread

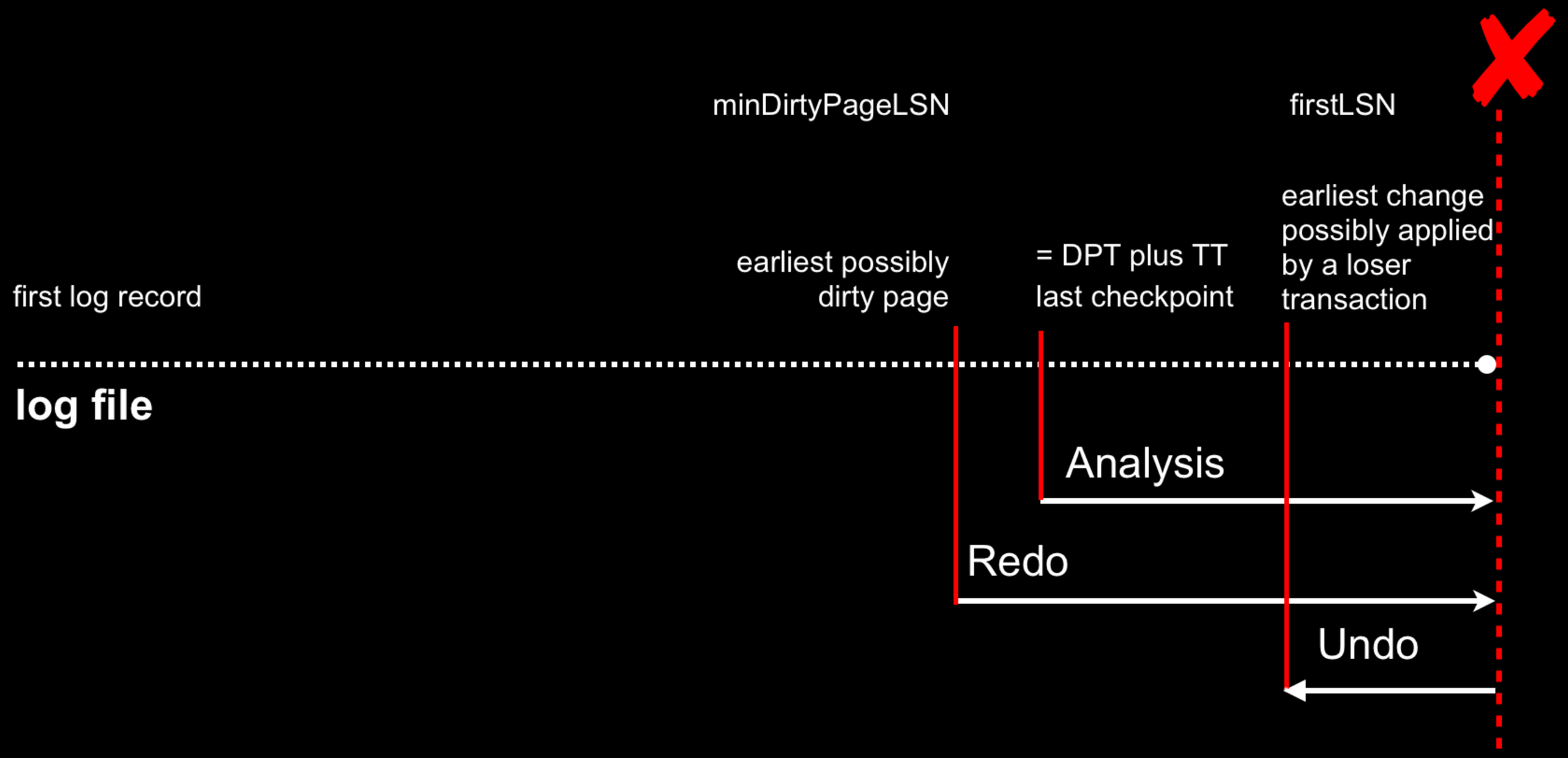
minDirtyPageLSN := SELECT min(recoveryLSN) FROM DPT;




note: relative starting points of the phases may differ

# Short Running Transactions

minDirtyPageLSN := SELECT min(recoveryLSN) FROM DPT;




note: relative starting points of the phases may differ

# Fuzzy Checkpointing Details

V473
TT + DPT
as of
LSN 473

Vnow()
TT + DPT
as of
Now()

COW

LSN=473

= DPT plus TT
as of begin_checkpoint

first log record

(1.)
begin_checkpoint

(2.)
end_checkpoint

last log record

buffer

**log file**

DPT+TT

Analysis

master record

(3.)
flush position of begin_checkpoint
to extra place on stable storage

# Tasks of the 3 Phases in ARIES

Analysis Phase:

construct DPT and TT to reflect state at the time of the crash

compute minDirtyPageLSN (where to start Redo phase)

Redo Phase: repeating history paradigm

repeat **all** actions: including those from loser transactions, including CLRs

# Tasks of the 3 Phases in ARIES

Analysis Phase:

- construct DPT and TT to reflect state at the time of the crash
- compute minDirtyPageLSN (where to start Redo phase)

Redo Phase:

- repeat **all** actions: including those from loser transactions, including CLRs
- restore database to where it was at the time of the crash

Undo Phase:

- undo actions of all loser transactions
- log undos to CLRs