Database Technology

Exercise 9 - Extended

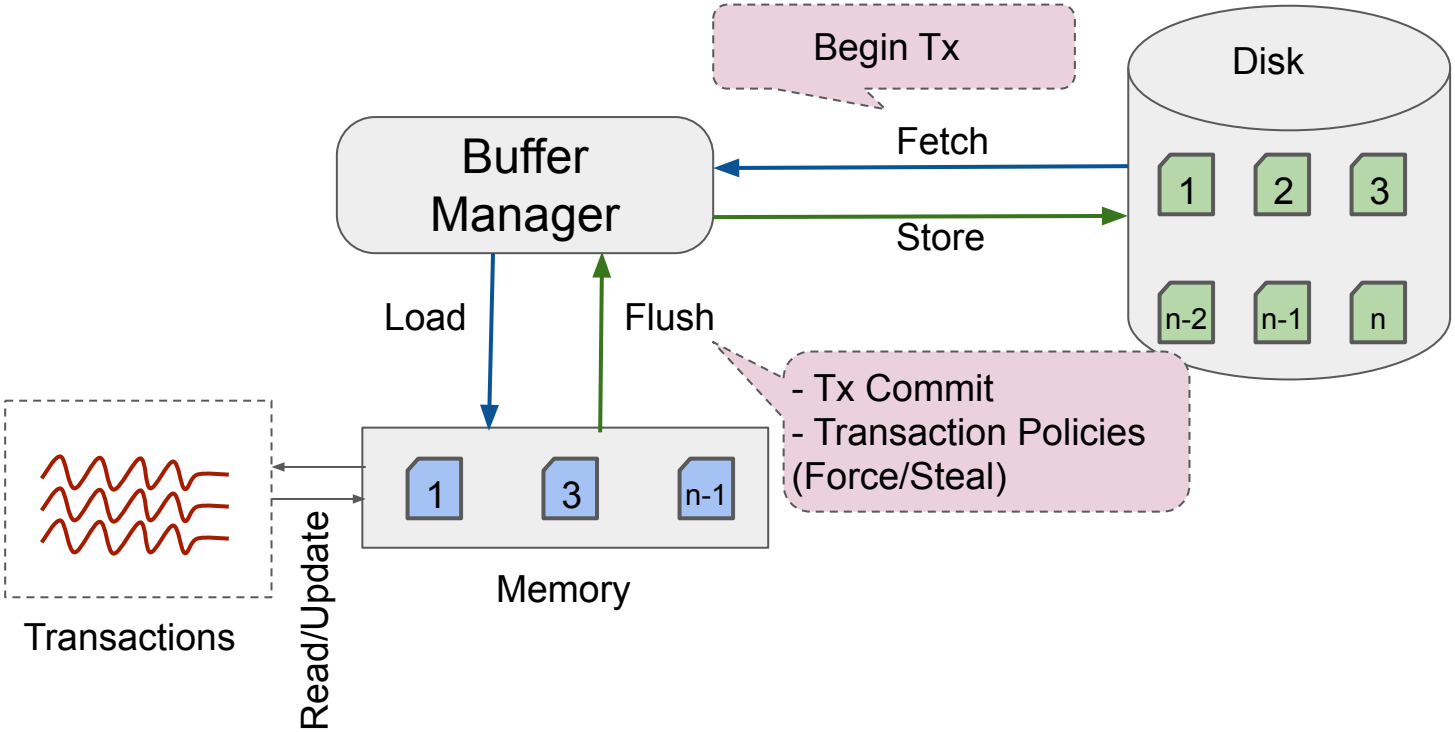
Why Do We Need Recovery?

- **Recovery:** What if the system fails in the middle of some execution (transactions, flush updated pages to disk)?
- **It ensures** database **consistency**, **atomicity**, and **durability** despite failures, i.e., it attempts to reconstruct a state of the data after a failure

Types of Failure

- **Transaction failure:**
  - Transaction aborts due to transaction code (e.g. division by 0.0, space allocation failure), abort command, deadlock resolution
- **Media failure:**
  - Mechanical hard disk failure, network partitioning.
- **System failure:**
  - Power or semiconductor failure, programming error, operating system crash. Memory lost but disk contents still persistent

Database Under Operation



Issues for Recovery

- **STEAL**
  - Mid-transaction state may be written to disk (due to buffer replacement)
  - Need to undo such actions after a crash

- **NO FORCE**
  - State of committed TXs may not be persisted, and thus lost at crash
  - Need to redo such actions

Write-Ahead Logging (WAL)

- **Force log record** for an update **before data page is written** to disk
  - Ensures atomicity, solves STEAL:
    Even if dirty state stable, log contains consistent state
- **Write all records** of a TX **before** TX commits
  - Ensures durability, solves NO FORCE:
    Even if TX changes not stable at crash, log describing changes stable

Log Records

- Every log record contains a **Log Sequence Number** (LSN)
- **Data page contains** PID and page LSN (pageLSN)
  - The LSN of the last log record that updates the page
- **Log record:**
  - Type, Transaction ID (TID), Previous LSN (prevLSN) (LSN of previous log record of same TX)

Recovery Using WAL

1. **Analysis pass**
   - Figure out which TX have been committed, aborted, or failed
   - Plus build some data structures for performance (Dirty and Transaction Tables)
2. **Redo pass (using Dirty Table)**
   - Read the log forward and redo all actions: “Repeating history” including for uncommitted transactions (read Mohan’s paper*).
   - Restores the database to a state where all TXs changes have been made stable
3. **Undo pass (using Transaction Table)**
   - Read the log backwards for uncommitted TXs only
   - Undo actions by failed TXs

** C. Mohan, Don Haderle, Bruce Lindsay, Hamid Pirahesh, and Peter Schwarz. 1992. ARIES: a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. ACM Trans. Database Syst. 17, 1 (March 1992), 94–162. https://doi.org/10.1145/128765.128770*

Checkpointing

- **Starting at the beginning** of the log every time is **not practical**
  - Need to consolidate changes periodically

- **Periodical Checkpoint:** Can start recovery from here
  - Write begin checkpoint log record (BCKPT)
    …
  - Write end checkpoint log record (ECKPT) that contains
    - Transaction table
    - Dirty Page Table
    - Analysis can start with saved DPT, TT that summarize history before BCKPT

Let's Build WAL

Write Ahead Logging (WAL)

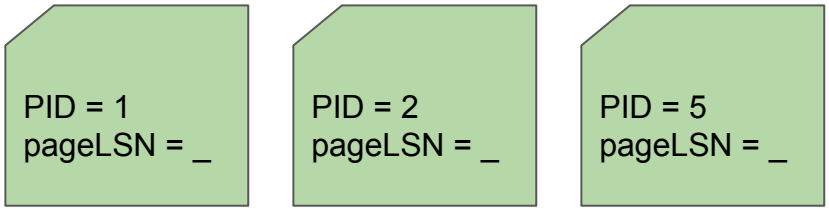
LSN	TID	prevLSN	Type

Transaction Table (TT)

TID	lastLSN

Dirty Page Table (DPT)

PID	recLSN



Let's Build WAL

Write Ahead Logging (WAL)

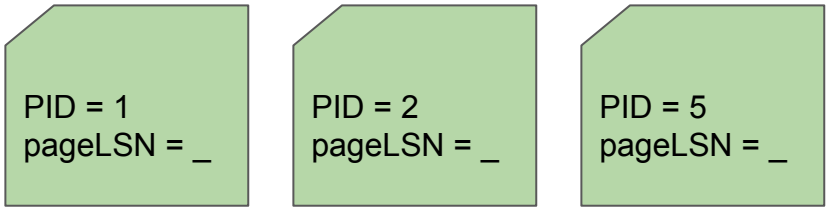
LSN	TID	prevLSN	Type
1	1		BEGIN_TX

Transaction Table (TT)

TID	lastLSN
1	**1**

Dirty Page Table (DPT)

PID	recLSN



Let's Build WAL

Write Ahead Logging (WAL)

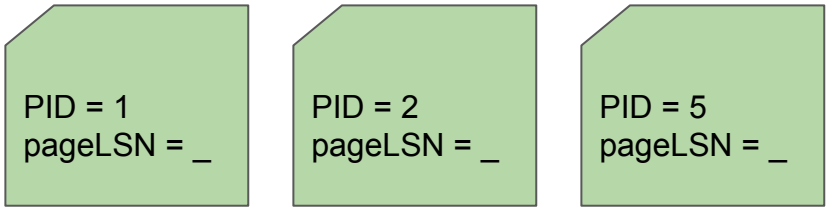
LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX

Transaction Table (TT)

TID	lastLSN
1	1
2	**2**

Dirty Page Table (DPT)

PID	recLSN



Let's Build WAL

Write Ahead Logging (WAL)

LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)

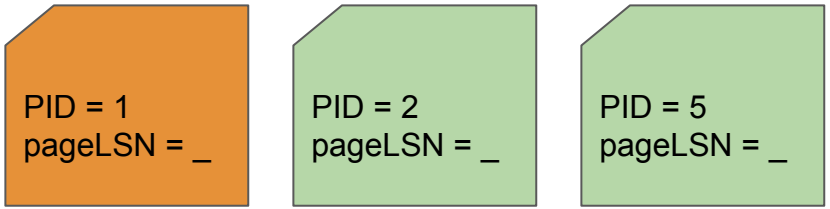
Dirtied Page 1

Transaction Table (TT)

TID	lastLSN
1	**3**
2	2

Dirty Page Table (DPT)

PID	recLSN
1	**3**



Let's Build WAL

Write Ahead Logging (WAL)

LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)

Dirtied Page 1

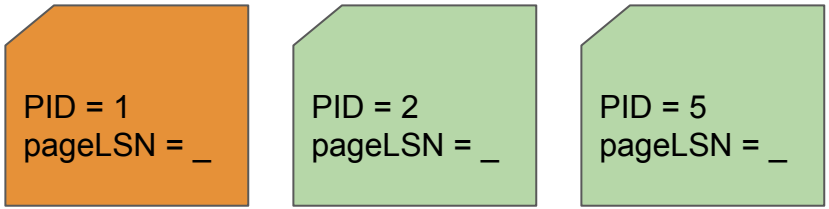
Transaction Table (TT)

TID	lastLSN
1	**3**
2	2

Dirty Page Table (DPT)

PID	recLSN
1	**3**

Page 1 is manipulated in memory and thus dirtied



Let's Build WAL

Write Ahead Logging (WAL)

LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)
4	2	2	UPDATE(2,…)

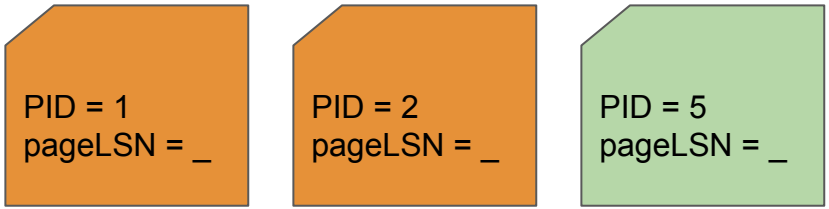
Dirtied Page 2

Transaction Table (TT)

TID	lastLSN
1	3
2	**4**

Dirty Page Table (DPT)

PID	recLSN
1	3
2	**4**



Let's Build WAL

Write Ahead Logging (WAL)

LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)
4	2	2	UPDATE(2,…)
5			BCHKPT

Transaction Table (TT)

TID	lastLSN
1	3
2	

Dirty Page Table (DPT)

D	recLSN
1	3
2	4

Checkpointed

PID = 1
pageLSN = _

PID = 2
pageLSN = _

PID = 5
pageLSN = _

Let's Build WAL

Write Ahead Logging (WAL)

LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)
4	2	2	UPDATE(2,…)
5			BCHKPT

Transaction Table (TT)

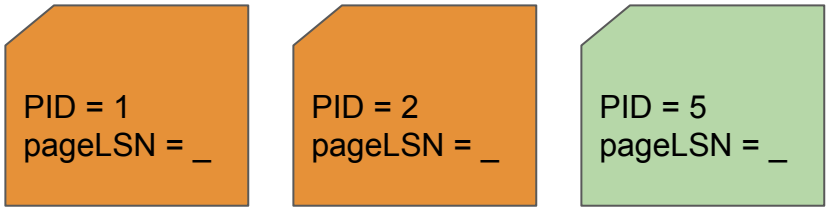
TID	lastLSN
1	3
2	

Dirty Page Table (DPT)

D	recLSN
1	3
2	4

Checkpointed

What if no checkpoint and there is a failure?



Let's Build WAL

Write Ahead Logging (WAL)

LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)
4	2	2	UPDATE(2,…)
5			BCHKPT

Transaction Table (TT)

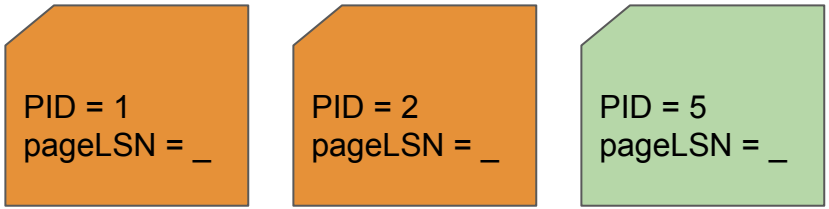
TID	lastLSN
1	3
2	

Dirty Page Table (DPT)

D	recLSN
1	3
2	4

Checkpointed

We need to re-compute TT and DPT by performing analysis pass over whole WAL without checkpointing.



Let's Build WAL

Write Ahead Logging (WAL)

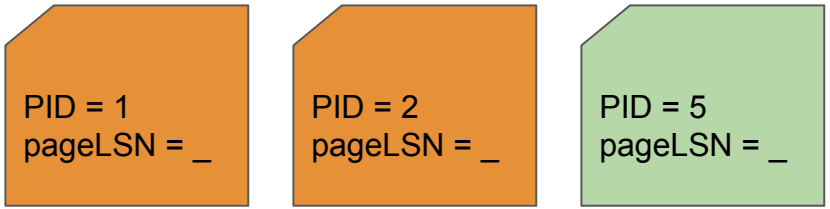
LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)
4	2	2	UPDATE(2,…)
5			BCHKPT
6	3		BEGIN_TX

Transaction Table (TT)

TID	lastLSN
1	3
2	4
3	**6**

Dirty Page Table (DPT)

PID	recLSN
1	3
2	4



Let's Build WAL

Write Ahead Logging (WAL)

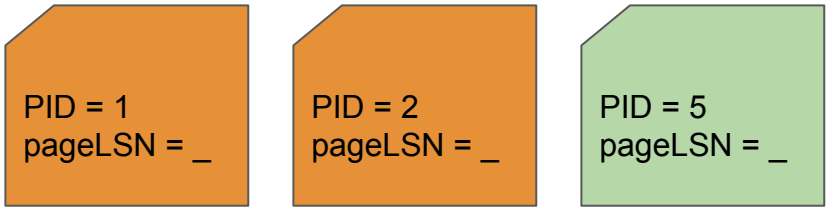
LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)
4	2	2	UPDATE(2,…)
5			BCHKPT
6	3		BEGIN_TX
7	2	5	COMMIT

Transaction Table (TT)

TID	lastLSN
1	3
2	**7**
3	6

Dirty Page Table (DPT)

PID	recLSN
1	3
2	4



Let's Build WAL

Write Ahead Logging (WAL)

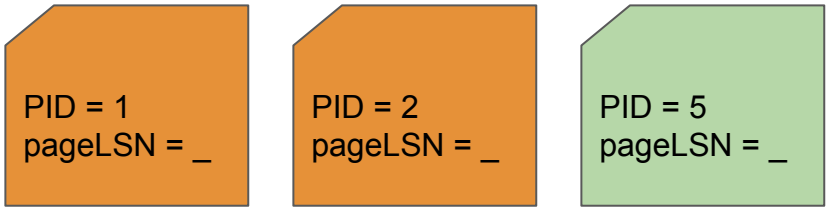
LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)
4	2	2	UPDATE(2,…)
5			BCHKPT
6	3		BEGIN_TX
7	2	5	COMMIT
8	2	8	END_TX

Transaction Table (TT)

TID	lastLSN
1	3
~~2~~	~~7~~
3	6

Dirty Page Table (DPT)

PID	recLSN
1	3
2	4



Let's Build WAL

Write Ahead Logging (WAL)

LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)
4	2	2	UPDATE(2,…)
5			BCHKPT
6	3		BEGIN_TX
7	2	5	COMMIT
8	2	8	END_TX
Flush Page 1 and 2 to Disk			

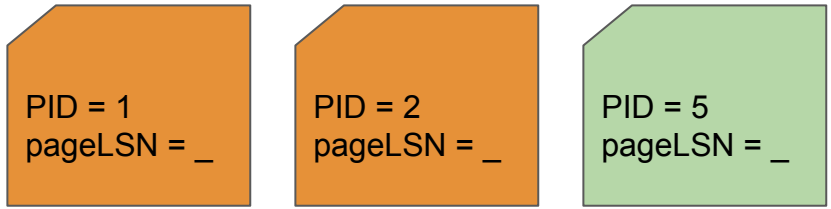
Transaction Table (TT)

TID	lastLSN
1	3
3	6

Dirty Page Table (DPT)

PID	recLSN
1	3
2	4

This is done in the background



Let's Build WAL

Write Ahead Logging (WAL)

LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)
4	2	2	UPDATE(2,…)
5			BCHKPT
6	3		BEGIN_TX
7	2	5	COMMIT
8	2	8	END_TX
Flush Page 1 and 2 to Disk			

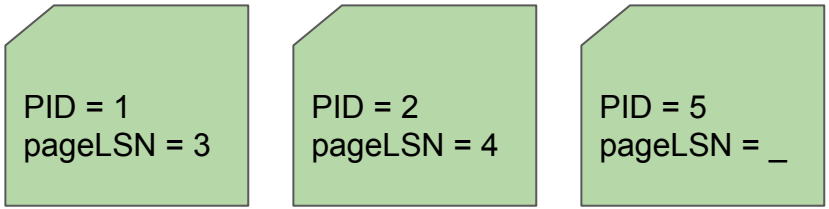
Transaction Table (TT)

TID	lastLSN
1	3
3	6

Dirty Page Table (DPT)

PID	recLSN
1	~~3~~
~~2~~	4

Page 1 & 2 are not dirty as they are flushed to disk



Let's Build WAL

Write Ahead Logging (WAL)

LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)
4	2	2	UPDATE(2,…)
5			BCHKPT
6	3		BEGIN_TX
7	2	5	COMMIT
8	2	8	END_TX
Flush Page 1 and 2 to Disk			
9			ECHKPT

Transaction Table (TT)

TID	lastLSN
1	3
3	6

Dirty Page Table (DPT)

PID	recLSN

PID = 1
pageLSN = 3

PID = 2
pageLSN = 4

PID = 5
pageLSN = _

Let's Build WAL

Write Ahead Logging (WAL)

LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)
4	2	2	UPDATE(2,…)
5			BCHKPT
6	3		BEGIN_TX
7	2	5	COMMIT
8	2	8	END_TX
Flush Page 1 and 2 to Disk			
9			ECHKPT
10	3	6	UPDATE(1,...)

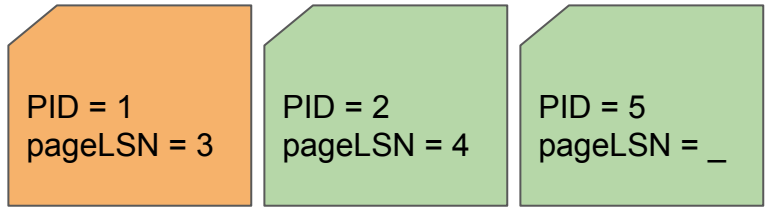
Dirtied Page 1

Transaction Table (TT)

TID	lastLSN
1	3
3	**10**

Dirty Page Table (DPT)

PID	recLSN
1	**10**



Let's Build WAL

Write Ahead Logging (WAL)

LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)
4	2	2	UPDATE(2,…)
5			BCHKPT
6	3		BEGIN_TX
7	2	5	COMMIT
8	2	8	END_TX
Flush Page 1 and 2 to Disk			
9			ECHKPT
10	3	6	UPDATE(1,...)
11	3	10	UPDATE(5,...)

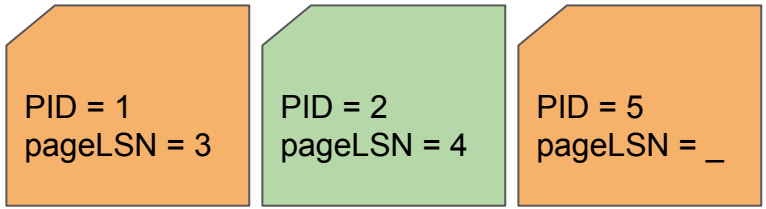
Dirtied Page 5

Transaction Table (TT)

TID	lastLSN
1	3
3	**11**

Dirty Page Table (DPT)

PID	recLSN
1	10
5	**11**



Let's Build WAL

Write Ahead Logging (WAL)

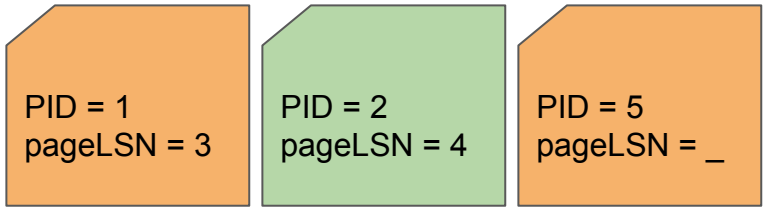
LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)
4	2	2	UPDATE(2,…)
5			BCHKPT
6	3		BEGIN_TX
7	2	5	COMMIT
8	2	8	END_TX
Flush Page 1 and 2 to Disk			
9			ECHKPT
10	3	6	UPDATE(1,...)
11	3	10	UPDATE(5,...)
12	1	3	READ

Transaction Table (TT)

TID	lastLSN
1	**12**
3	11

Dirty Page Table (DPT)

PID	recLSN
1	10
5	11



That’s it folks! You now know:

- Need for recovery
- Role of logs
- Why checkpointing?
- How WAL is built?
- How to do Analysis Pass?

What about Recover?

Redo

Repeat the history by reading the WAL

- **Scan log forward**
  1. From smallest recLSN in DPT
  2. Read PID from log record
  3. **If** *(PID not in DPT)* **or** *(PID in DPT And recLSN > LSN)* **do**
  4. **Nothing** (Why? When does this happen?)
  5. **Else**
  6. Fetch page with PID from disk
  7. **If** *pageLSN >= LSN* **do**
  8. **Nothing**
  9. **Else**
  10. **Redo**
- **Redoing a page:** *Re-apply logged action and Set pageLSN = LSN*

Let's Perform Redo Pass From LSN 3

Write Ahead Logging (WAL)

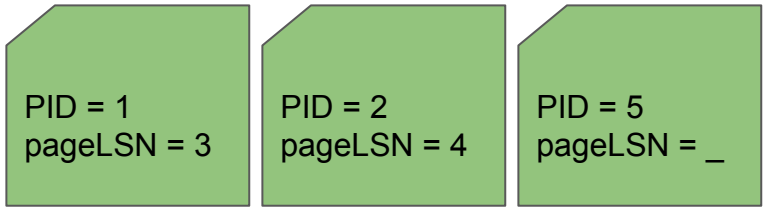
LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)
4	2	2	UPDATE(2,…)
5			BCHKPT
6	3		BEGIN_TX
7	2	5	COMMIT
8	2	8	END_TX
Flush Page 1 and 2 to Disk			
9			ECHKPT
10	3	7	UPDATE(1,...)
11	3	10	UPDATE(5,...)
12	1	3	READ

Redo

Dirty Page Table (DPT)

PID	recLSN
1	10
5	11

LSN=3, PID = 1
PID exists in DPT with recLSN =10
Do nothing line 3-4



Let's Perform Redo Pass From LSN 3

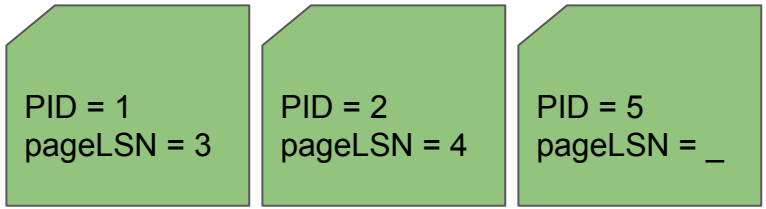
Write Ahead Logging (WAL)

LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)
4	2	2	UPDATE(2,…)
5			BCHKPT
6	3		BEGIN_TX
7	2	5	COMMIT
8	2	8	END_TX
Flush Page 1 and 2 to Disk			
9			ECHKPT
10	3	7	UPDATE(1,...)
11	3	10	UPDATE(5,...)
12	1	3	READ

Dirty Page Table (DPT)

PID	recLSN
1	10
5	11

LSN=4, PID = 2
PID does not exists in DPT
Do nothing line 3-4



Let's Perform Redo Pass From LSN 3

Write Ahead Logging (WAL)

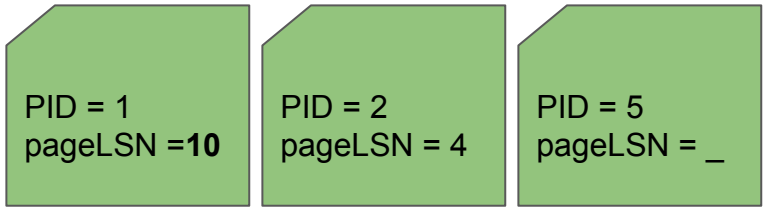
LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)
4	2	2	UPDATE(2,…)
5			BCHKPT
6	3		BEGIN_TX
7	2	5	COMMIT
8	2	8	END_TX
Flush Page 1 and 2 to Disk			
9			ECHKPT
10	3	7	UPDATE(1,...)
11	3	10	UPDATE(5,...)
12	1	3	READ

Redo

Dirty Page Table (DPT)

PID	recLSN
1	10
5	11

LSN=10, PID = 1
PID exists in DPT with recLSN = 10
Condition on line 3 fails
Fetch page from disk with PID 1 pageLSN = 3
Condition on line 7 fails
Perform Redo



Let's Perform Redo Pass From LSN 3

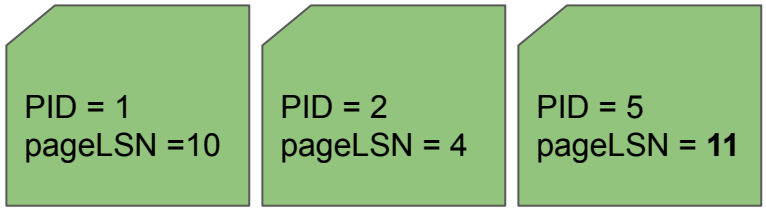
Write Ahead Logging (WAL)

LSN	TID	prevLSN	Type	
1	1		BEGIN_TX	
2	2		BEGIN_TX	
3	1	1	UPDATE(1,…)	⬅
4	2	2	UPDATE(2,…)	⬅
5			BCHKPT	
6	3		BEGIN_TX	
7	2	5	COMMIT	
8	2	8	END_TX	
Flush Page 1 and 2 to Disk				
9			ECHKPT	
10	3	7	UPDATE(1,...)	⬅ Redo
11	3	10	UPDATE(5,...)	⬅ **Redo**
12	1	3	READ	

Dirty Page Table (DPT)

PID	recLSN
1	10
5	11

LSN=11, PID = 5
PID exists in DPT with recLSN = 11
Condition on line 3 fails
Fetch page from disk with PID 5 pageLSN = _
Condition on line 7 fails
Perform Redo



Undo

- Scan log **backwards** for “loser” TXs
  1. ToUndo **=** {lastLSN of loser TXs}
  2. Choose largest LSN from ToUndo
  3. **If** (log record is update) **do**
  4. Undo
  5. **add** prevLSN **to** ToUndo
- No need to consult DPT, pageLSN
- We are logging during undo
- Undo has the same logic as transaction rollback
- It can be done in parallel for each TX, database can be available during undo

Let's Undo the Changes

Write Ahead Logging (WAL)

LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)
4	2	2	UPDATE(2,…)
5			BCHKPT
6	3		BEGIN_TX
7	2	5	COMMIT
8	2	8	END_TX
Flush Page 1 and 2 to Disk			
9			ECHKPT
10	3	7	UPDATE(1,...)
11	3	10	UPDATE(5,...)
12	1	3	READ

⬅ **Undo**

Transaction Table (TT)

TID	lastLSN
1	12
3	11

Fetch all recLSNs from TT to ToUndo: {**12**,11}

Let's Undo the Changes

Write Ahead Logging (WAL)

LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)
4	2	2	UPDATE(2,…)
5			BCHKPT
6	3		BEGIN_TX
7	2	5	COMMIT
8	2	8	END_TX
Flush Page 1 and 2 to Disk			
9			ECHKPT
10	3	7	UPDATE(1,...)
11	3	10	UPDATE(5,...)
12	1	3	READ

Undo

Transaction Table (TT)

TID	lastLSN
1	12
3	11

Fetch all recLSNs from TT to ToUndo: {**11**,3}

Compensatory Log Record (CRL) is added in WAL (not shown here)

Let's Undo the Changes

Write Ahead Logging (WAL)

LSN	TID	prevLSN	Type
1	1		BEGIN_TX
2	2		BEGIN_TX
3	1	1	UPDATE(1,…)
4	2	2	UPDATE(2,…)
5			BCHKPT
6	3		BEGIN_TX
7	2	5	COMMIT
8	2	8	END_TX
Flush Page 1 and 2 to Disk			
9			ECHKPT
10	3	7	UPDATE(1,...)
11	3	10	UPDATE(5,...)
12	1	3	READ

Undo

Undo

Transaction Table (TT)

TID	lastLSN
1	3
3	11

Fetch all recLSNs from TT to ToUndo: {**10**,3}

Let's Undo the Changes

Write Ahead Logging (WAL)

LSN	TID	prevLSN	Type	
1	1		BEGIN_TX	
2	2		BEGIN_TX	
3	1	1	UPDATE(1,…)	
4	2	2	UPDATE(2,…)	
5			BCHKPT	
6	3		BEGIN_TX	⬅
7	2	5	COMMIT	
8	2	8	END_TX	
Flush Page 1 and 2 to Disk				
9			ECHKPT	
10	3	6	~~UPDATE(1,...)~~	⬅ Undo
11	3	10	~~UPDATE(5,...)~~	⬅ Undo
12	1	3	READ	

Transaction Table (TT)

TID	lastLSN
1	3
3	11

Fetch all recLSNs from TT to ToUndo: {**6**,3}

Let's Undo the Changes

Write Ahead Logging (WAL)

LSN	TID	prevLSN	Type	
1	1		BEGIN_TX	
2	2		BEGIN_TX	
3	1	1	UPDATE(1,…)	← Undo
4	2	2	UPDATE(2,…)	
5			BCHKPT	
6	3		BEGIN_TX	←
7	2	5	COMMIT	
8	2	8	END_TX	
Flush Page 1 and 2 to Disk				
9			ECHKPT	
10	3	6	~~UPDATE(1,...)~~	← Undo
11	3	10	~~UPDATE(5,...)~~	← Undo
12	1	3	READ	

Transaction Table (TT)

TID	lastLSN
1	3
3	11

Fetch all recLSNs from TT to ToUndo: {**3**}

Let's Undo the Changes

Write Ahead Logging (WAL)

LSN	TID	prevLSN	Type	
1	1		BEGIN_TX	
2	2		BEGIN_TX	
3	1	1	~~UPDATE(1,…)~~	Undo
4	2	2	UPDATE(2,…)	
5			BCHKPT	
6	3		BEGIN_TX	
7	2	5	COMMIT	
8	2	8	END_TX	
Flush Page 1 and 2 to Disk				
9			ECHKPT	
10	3	6	~~UPDATE(1,...)~~	Undo
11	3	10	~~UPDATE(5,...)~~	Undo
12	1	3	READ	

Transaction Table (TT)

TID	lastLSN
1	3
3	11

Fetch all recLSNs from TT to ToUndo: {**1**}

Analysis Pass

Variant 1

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION
1	1		BEGIN_TRANSACTION
2	2		BEGIN_TRANSACTION
3	3		BEGIN_TRANSACTION
4	1	1	UPDATE(1, 1, "a", "A")
5	2	2	UPDATE(2, 2, "b", "B")
6	1	4	UPDATE (1, 3, "c", "C")
7			BEGIN_CHECKPOINT
8th	1	6	UPDATE (1, 4, "d", "D")
9	2	5	UPDATE (3, 5, "e", "E")
10	2	9	COMMIT
11	2	10	END_TRANSACTION
12			END_CHECKPOINT
13	3	3	UPDATE (2, 6, "f", "F")
14	3	13	UPDATE (3, 7, "g", "G")

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	6
2	5
3	3

Dirty page table:

PID	LSN
1	4
2	5

In the transaction and dirty page tables below, enter the LSN for each transaction/page after the analysis pass.
Select "No Entry", if a transaction/page is not in the transaction/dirty page table.

Transaction table:

TID	LSN
1	8
2	No Entry
3	14

Dirty page table:

PID	LSN
1	4
2	5
3	9

Variant 2

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION
1	1		BEGIN_TRANSACTION
2	2		BEGIN_TRANSACTION
3	3		BEGIN_TRANSACTION
4	1	1	UPDATE(1, 1, "a", "A")
5	2	2	UPDATE(2, 2, "b", "B")
6	1	4	UPDATE (2, 3, "c", "C")
7			BEGIN_CHECKPOINT
8th	1	6	UPDATE (3, 4, "d", "D")
9	3	3	UPDATE (2, 5, "e", "E")
10	1	8th	COMMIT
11	1	10	END_TRANSACTION
12			END_CHECKPOINT
13	2	5	UPDATE (3, 6, "f", "F")
14	2	13	UPDATE (1, 7, "g", "G")

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	6
2	5
3	3

Dirty page table:

PID	LSN
1	4
2	5

In the transaction and dirty page tables below, enter the LSN for each transaction/page after the analysis pass.
Select "No Entry", if a transaction/page is not in the transaction/dirty page table.

Transaction table:

TID	LSN
1	No Entry
2	14
3	9

Dirty page table:

PID	LSN
1	4
2	5
3	8

Variant 3

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION
1	1		BEGIN_TRANSACTION
2	2		BEGIN_TRANSACTION
3	3		BEGIN_TRANSACTION
4	1	1	UPDATE(1, 1, "a", "A")
5	2	2	UPDATE(2, 2, "b", "B")
6	2	5	UPDATE (1, 3, "c", "C")
7			BEGIN_CHECKPOINT
8th	3	3	UPDATE (3, 4, "d", "D")
9	3	8th	UPDATE (3, 5, "e", "E")
10	3	9	COMMIT
11	3	10	END_TRANSACTION
12			END_CHECKPOINT
13	1	4	UPDATE (3, 6, "f", "F")
14	1	13	UPDATE (2, 7, "g", "G")

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	4
2	6
3	3

Dirty page table:

PID	LSN
1	4
2	5

In the transaction and dirty page tables below, enter the LSN for each transaction/page after the analysis pass.
Select "No Entry", if a transaction/page is not in the transaction/dirty page table.

Transaction table:

TID	LSN
1	14
2	6
3	No Entry

Dirty page table:

PID	LSN
1	4
2	5
3	8

Variant 4

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry.
The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION
1	1		BEGIN_TRANSACTION
2	2		BEGIN_TRANSACTION
3	3		BEGIN_TRANSACTION
4	1	1	UPDATE(3, 1, "a", "A")
5	2	2	UPDATE(3, 2, "b", "B")
6	2	5	UPDATE (2, 3, "c", "C")
7			BEGIN_CHECKPOINT
8th	1	4	UPDATE (2, 4, "d", "D")
9	3	3	UPDATE (1, 5, "e", "E")
10	1	8th	COMMIT
11	1	10	END_TRANSACTION
12			END_CHECKPOINT
13	2	6	UPDATE (3, 6, "f", "F")
14	2	13	UPDATE (2, 7, "g", "G")

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	4
2	6
3	3

Dirty page table:

PID	LSN
2	6
3	4

In the transaction and dirty page tables below, enter the LSN for each transaction/page after the analysis pass.
Select "No Entry", if a transaction/page is not in the transaction/dirty page table.

Transaction table:

TID	LSN
1	No Entry
2	14
3	9

Dirty page table:

PID	LSN
1	9
2	6
3	4

Variant 5

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION
1	1		BEGIN_TRANSACTION
2	2		BEGIN_TRANSACTION
3	3		BEGIN_TRANSACTION
4	1	1	UPDATE(3, 1, "a", "A")
5	3	3	UPDATE(3, 2, "b", "B")
6	2	2	UPDATE (3, 3, "c", "C")
7			BEGIN_CHECKPOINT
8th	1	4	UPDATE (2, 4, "d", "D")
9	3	5	UPDATE (1, 5, "e", "E")
10	1	8th	COMMIT
11	1	10	END_TRANSACTION
12			END_CHECKPOINT
13	2	6	UPDATE (3, 6, "f", "F")
14	2	13	UPDATE (2, 7, "g", "G")

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	4
2	6
3	5

Dirty page table:

PID	LSN
3	4

In the transaction and dirty page tables below, enter the LSN for each transaction/page after the analysis pass.
Select "No Entry", if a transaction/page is not in the transaction/dirty page table.

Transaction table:

TID	LSN
1	No Entry
2	14
3	9

Dirty page table:

PID	LSN
1	9
2	8
3	4

Variant 6

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION
1	1		BEGIN_TRANSACTION
2	2		BEGIN_TRANSACTION
3	3		BEGIN_TRANSACTION
4	1	1	UPDATE(3, 1, "a", "A")
5	3	3	UPDATE(3, 2, "b", "B")
6	2	2	UPDATE (3, 3, "c", "C")
7			BEGIN_CHECKPOINT
8th	3	5	UPDATE (1, 4, "d", "D")
9	2	6	UPDATE (1, 5, "e", "E")
10	1	4	COMMIT
11	1	10	END_TRANSACTION
12			END_CHECKPOINT
13	2	9	UPDATE (2, 6, "f", "F")
14	2	13	UPDATE (2, 7, "g", "G")

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	4
2	6
3	5

Dirty page table:

PID	LSN
3	4

In the transaction and dirty page tables below, enter the LSN for each transaction/page after the analysis pass.
Select "No Entry", if a transaction/page is not in the transaction/dirty page table.

Transaction table:

TID	LSN
1	No Entry
2	14
3	8

Dirty page table:

PID	LSN
1	8
2	13
3	4

Redo Pass

Variant 1

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION	Redo Pass
1	1		BEGIN_TRANSACTION	
2	2		BEGIN_TRANSACTION	
3	3		BEGIN_TRANSACTION	
4	1	1	UPDATE (1, 1, "a", "A")	Redo
5	2	2	UPDATE (2, 2, "b", "B")	Redo
6	1	4	UPDATE (1, 3, "c", "C")	Redo
7			BEGIN_CHECKPOINT	
8	1	6	UPDATE (1, 4, "d", "D")	Redo
9	2	5	UPDATE (3, 5, "e", "E")	Redo
10	2	9	COMMIT	
11	2	10	END_TRANSACTION	
12			END_CHECKPOINT	
13	1	8	UPDATE (2, 6, "f", "F")	
14	3	3	UPDATE (3, 7, "g", "G")	

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	6
2	5
3	3

Dirty page table:

PID	LSN
1	4
2	5

Select the action taken during the redo passes for the UPDATE actions in the drop-down menu in the log above.
(Assume that pageLSN is always less than LSN).

Variant 2

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION	Redo Pass
1	1		BEGIN_TRANSACTION	
2	2		BEGIN_TRANSACTION	
3	3		BEGIN_TRANSACTION	
4	1	1	UPDATE (1, 1, "a", "A")	Ignore
5	2	2	UPDATE (2, 2, "b", "B")	Redo
6	1	4	UPDATE (1, 3, "c", "C")	Redo
7			BEGIN_CHECKPOINT	
8	1	6	UPDATE (1, 4, "d", "D")	Redo
9	2	5	UPDATE (3, 5, "e", "E")	Redo
10	2	9	COMMIT	
11	2	10	END_TRANSACTION	
12			END_CHECKPOINT	
13	1	8	UPDATE (2, 6, "f", "F")	
14	3	3	UPDATE (3, 7, "g", "G")	

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	6
2	5
3	3

Dirty page table:

PID	LSN
1	6
2	5

Select the action taken during the redo passes for the UPDATE actions in the drop-down menu in the log above.
(Assume that pageLSN is always less than LSN).

Variant 3

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION	Redo Pass
1	1		BEGIN_TRANSACTION	
2	2		BEGIN_TRANSACTION	
3	3		BEGIN_TRANSACTION	
4	1	1	UPDATE (1, 1, "a", "A")	Ignore
5	2	2	UPDATE (2, 2, "b", "B")	Ignore
6	1	4	UPDATE (1, 3, "c", "C")	Redo
7			BEGIN_CHECKPOINT	
8	1	6	UPDATE (1, 4, "d", "D")	Redo
9	2	5	UPDATE (3, 5, "e", "E")	Redo
10	2	9	COMMIT	
11	2	10	END_TRANSACTION	
12			END_CHECKPOINT	
13	1	8	UPDATE (2, 6, "f", "F")	
14	3	3	UPDATE (3, 7, "g", "G")	

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	6
2	5
3	3

Dirty page table:

PID	LSN
1	6

Select the action taken during the redo passes for the UPDATE actions in the drop-down menu in the log above.
(Assume that pageLSN is always less than LSN).

Variant 4

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION	Redo Pass
1	1		BEGIN_TRANSACTION	
2	2		BEGIN_TRANSACTION	
3	3		BEGIN_TRANSACTION	
4	1	1	UPDATE (1, 1, "a", "A")	Ignore
5	2	2	UPDATE (2, 2, "b", "B")	Ignore
6	1	4	UPDATE (1, 3, "c", "C")	Ignore
7			BEGIN_CHECKPOINT	
8	1	6	UPDATE (1, 4, "d", "D")	Redo
9	2	5	UPDATE (3, 5, "e", "E")	Redo
10	2	9	COMMIT	
11	2	10	END_TRANSACTION	
12			END_CHECKPOINT	
13	1	8	UPDATE (2, 6, "f", "F")	
14	3	3	UPDATE (3, 7, "g", "G")	

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	6
2	5
3	3

Dirty page table:

PID	LSN
1	8

Select the action taken during the redo passes for the UPDATE actions in the drop-down menu in the log above.
(Assume that pageLSN is always less than LSN).

Variant 5

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION	Redo Pass
1	1		BEGIN_TRANSACTION	
2	2		BEGIN_TRANSACTION	
3	3		BEGIN_TRANSACTION	
4	1	1	UPDATE (1, 1, "a", "A")	Redo
5	2	2	UPDATE (2, 2, "b", "B")	Redo
6	2	5	UPDATE (1, 3, "c", "C")	Redo
7			BEGIN_CHECKPOINT	
8	2	6	UPDATE (1, 4, "d", "D")	Redo
9	2	8	UPDATE (3, 5, "e", "E")	Redo
10	2	9	COMMIT	
11	2	10	END_TRANSACTION	
12			END_CHECKPOINT	
13	3	3	UPDATE (2, 6, "f", "F")	
14	3	13	UPDATE (3, 7, "g", "G")	

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	4
2	6
3	3

Dirty page table:

PID	LSN
1	4
2	5

Select the action taken during the redo passes for the UPDATE actions in the drop-down menus in the log above.
(Assume that pageLSN is always less than LSN).

Variant 6

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION	Redo Pass
1	1		BEGIN_TRANSACTION	
2	2		BEGIN_TRANSACTION	
3	3		BEGIN_TRANSACTION	
4	1	1	UPDATE (1, 1, "a", "A")	Ignore
5	2	2	UPDATE (2, 2, "b", "B")	Redo
6	2	5	UPDATE (1, 3, "c", "C")	Redo
7			BEGIN_CHECKPOINT	
8	2	6	UPDATE (1, 4, "d", "D")	Redo
9	2	8	UPDATE (3, 5, "e", "E")	Redo
10	2	9	COMMIT	
11	2	10	END_TRANSACTION	
12			END_CHECKPOINT	
13	3	3	UPDATE (2, 6, "f", "F")	
14	3	13	UPDATE (3, 7, "g", "G")	

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	4
2	6
3	3

Dirty page table:

PID	LSN
1	6
2	5

Select the action taken during the redo passes for the UPDATE actions in the drop-down menus in the log above.
(Assume that pageLSN is always less than LSN).

Variant 7

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION	Redo Pass
1	1		BEGIN_TRANSACTION	
2	2		BEGIN_TRANSACTION	
3	3		BEGIN_TRANSACTION	
4	1	1	UPDATE (1, 1, "a", "A")	Ignore
5	2	2	UPDATE (2, 2, "b", "B")	Ignore
6	2	5	UPDATE (1, 3, "c", "C")	Redo
7			BEGIN_CHECKPOINT	
8	2	6	UPDATE (1, 4, "d", "D")	Redo
9	2	8	UPDATE (3, 5, "e", "E")	Redo
10	2	9	COMMIT	
11	2	10	END_TRANSACTION	
12			END_CHECKPOINT	
13	3	3	UPDATE (2, 6, "f", "F")	
14	3	13	UPDATE (3, 7, "g", "G")	

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	4
2	6
3	3

Dirty page table:

PID	LSN
1	6

Select the action taken during the redo passes for the UPDATE actions in the drop-down menus in the log above.
(Assume that pageLSN is always less than LSN).

Variant 8

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION	Redo Pass
1	1		BEGIN_TRANSACTION	
2	2		BEGIN_TRANSACTION	
3	3		BEGIN_TRANSACTION	
4	1	1	UPDATE (1, 1, "a", "A")	Ignore
5	2	2	UPDATE (2, 2, "b", "B")	Ignore
6	2	5	UPDATE (1, 3, "c", "C")	Ignore
7			BEGIN_CHECKPOINT	
8	2	6	UPDATE (1, 4, "d", "D")	Redo
9	2	8	UPDATE (3, 5, "e", "E")	Redo
10	2	9	COMMIT	
11	2	10	END_TRANSACTION	
12			END_CHECKPOINT	
13	3	3	UPDATE (2, 6, "f", "F")	
14	3	13	UPDATE (3, 7, "g", "G")	

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	4
2	6
3	3

Dirty page table:

PID	LSN
1	8

Select the action taken during the redo passes for the UPDATE actions in the drop-down menus in the log above.
(Assume that pageLSN is always less than LSN).

Undo Pass

Variant 1

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION	Undo Pass
1	1		BEGIN_TRANSACTION	
2	2		BEGIN_TRANSACTION	
3	3		BEGIN_TRANSACTION	
4	1	1	UPDATE (1, 1, "a", "A")	
5	2	2	UPDATE (2, 2, "b", "B")	
6	1	4	UPDATE (1, 3, "c", "C")	Undo
7			BEGIN_CHECKPOINT	
8	1	6	UPDATE (1, 4, "d", "D")	Undo
9	2	5	UPDATE (3, 5, "e", "E")	Ignore
10	2	9	COMMIT	
11	2	10	END_TRANSACTION	
12			END_CHECKPOINT	
13	3	3	UPDATE (2, 6, "f", "F")	Undo
14	3	13	UPDATE (3, 7, "g", "G")	Undo

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	6
2	5
3	3

Dirty page table:

PID	LSN
1	4
2	5

Select the action taken during the undo passes for the UPDATE actions in the drop-down menus in the log above.

Variant 2

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION	Undo Pass
1	1		BEGIN_TRANSACTION	
2	2		BEGIN_TRANSACTION	
3	3		BEGIN_TRANSACTION	
4	1	1	UPDATE (1, 1, "a", "A")	
5	2	2	UPDATE (2, 2, "b", "B")	
6	1	4	UPDATE (2, 3, "c", "C")	Ignore
7			BEGIN_CHECKPOINT	
8	1	6	UPDATE (3, 4, "d", "D")	Ignore
9	3	3	UPDATE (2, 5, "e", "E")	Undo
10	1	8	COMMIT	
11	1	10	END_TRANSACTION	
12			END_CHECKPOINT	
13	2	5	UPDATE (3, 6, "f", "F")	Undo
14	2	13	UPDATE (1, 7, "g", "G")	Undo

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	6
2	5
3	3

Dirty page table:

PID	LSN
1	4
2	5

Select the action taken during the undo passes for the UPDATE actions in the drop-down menus in the log above.

Variant 3

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION	Undo Pass
1	1		BEGIN_TRANSACTION	
2	2		BEGIN_TRANSACTION	
3	3		BEGIN_TRANSACTION	
4	1	1	UPDATE (1, 1, "a", "A")	
5	2	2	UPDATE (2, 2, "b", "B")	
6	2	5	UPDATE (1, 3, "c", "C")	Undo
7			BEGIN_CHECKPOINT	
8	3	3	UPDATE (3, 4, "d", "D")	Ignore
9	3	8	UPDATE (3, 5, "e", "E")	Ignore
10	3	9	COMMIT	
11	3	10	END_TRANSACTION	
12			END_CHECKPOINT	
13	1	4	UPDATE (3, 6, "f", "F")	Undo
14	1	13	UPDATE (2, 7, "g", "G")	Undo

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	4
2	6
3	3

Dirty page table:

PID	LSN
1	4
2	5

Select the action taken during the undo passes for the UPDATE actions in the drop-down menus in the log above.

Variant 4

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION	Undo Pass
1	1		BEGIN_TRANSACTION	
2	2		BEGIN_TRANSACTION	
3	3		BEGIN_TRANSACTION	
4	1	1	UPDATE (3, 1, "a", "A")	
5	2	2	UPDATE (3, 2, "b", "B")	
6	2	5	UPDATE (2, 3, "c", "C")	Undo
7			BEGIN_CHECKPOINT	
8	1	4	UPDATE (2, 4, "d", "D")	Ignore
9	3	3	UPDATE (1, 5, "e", "E")	Undo
10	1	8	COMMIT	
11	1	10	END_TRANSACTION	
12			END_CHECKPOINT	
13	2	6	UPDATE (3, 6, "f", "F")	Undo
14	2	13	UPDATE (2, 7, "g", "G")	Undo

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	4
2	6
3	3

Dirty page table:

PID	LSN
2	6
3	4

Select the action taken during the undo passes for the UPDATE actions in the drop-down menus in the log above.

Variant 5

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION	Undo Pass
1	1		BEGIN_TRANSACTION	
2	2		BEGIN_TRANSACTION	
3	3		BEGIN_TRANSACTION	
4	1	1	UPDATE (1, 1, "a", "A")	
5	2	2	UPDATE (2, 2, "b", "B")	
6	2	5	UPDATE (1, 3, "c", "C")	Ignore
7			BEGIN_CHECKPOINT	
8	1	4	UPDATE (1, 4, "d", "D")	Undo
9	2	6	UPDATE (3, 5, "e", "E")	Ignore
10	2	9	COMMIT	
11	2	10	END_TRANSACTION	
12			END_CHECKPOINT	
13	3	3	UPDATE (2, 6, "f", "F")	Undo
14	3	13	UPDATE (3, 7, "g", "G")	Undo

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	4
2	6
3	3

Dirty page table:

PID	LSN
1	4
2	5

Select the action taken during the undo passes for the UPDATE actions in the drop-down menus in the log above.

Variant 6

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION	Undo Pass
1	1		BEGIN_TRANSACTION	
2	2		BEGIN_TRANSACTION	
3	3		BEGIN_TRANSACTION	
4	1	1	UPDATE (1, 1, "a", "A")	
5	2	2	UPDATE (2, 2, "b", "B")	
6	2	5	UPDATE (1, 3, "c", "C")	Ignore
7			BEGIN_CHECKPOINT	
8	2	6	UPDATE (1, 4, "d", "D")	Ignore
9	2	8	UPDATE (3, 5, "e", "E")	Ignore
10	2	9	COMMIT	
11	2	10	END_TRANSACTION	
12			END_CHECKPOINT	
13	3	3	UPDATE (2, 6, "f", "F")	Undo
14	3	13	UPDATE (3, 7, "g", "G")	Undo

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	4
2	6
3	3

Dirty page table:

PID	LSN
1	4
2	5

Select the action taken during the undo passes for the UPDATE actions in the drop-down menus in the log above.

Variant 7

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION	Undo Pass
1	1		BEGIN_TRANSACTION	
2	2		BEGIN_TRANSACTION	
3	3		BEGIN_TRANSACTION	
4	1	1	UPDATE (1, 1, "a", "A")	
5	2	2	UPDATE (2, 2, "b", "B")	
6	1	4	UPDATE (2, 3, "c", "C")	Ignore
7			BEGIN_CHECKPOINT	
8	3	3	UPDATE (3, 4, "d", "D")	Undo
9	3	8	UPDATE (2, 5, "e", "E")	Undo
10	1	6	COMMIT	
11	1	10	END_TRANSACTION	
12			END_CHECKPOINT	
13	2	5	UPDATE (3, 6, "f", "F")	Undo
14	2	13	UPDATE (1, 7, "g", "G")	Undo

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	6
2	5
3	3

Dirty page table:

PID	LSN
1	4
2	5

Select the action taken during the undo passes for the UPDATE actions in the drop-down menus in the log above.

Variant 8

Consider the following log records. The table contains the log sequence number (LSN), the transaction ID (TID), the previous log sequence number of the transaction (PSN), and the action of the log entry. The format of the UPDATE action is: (Page ID, Element ID, Old value, New value).

LSN	TID	PSN	ACTION	Undo Pass
1	1		BEGIN_TRANSACTION	
2	2		BEGIN_TRANSACTION	
3	3		BEGIN_TRANSACTION	
4	1	1	UPDATE (1, 1, "a", "A")	
5	2	2	UPDATE (2, 2, "b", "B")	
6	2	5	UPDATE (1, 3, "c", "C")	Undo
7			BEGIN_CHECKPOINT	
8	3	3	UPDATE (3, 4, "d", "D")	Ignore
9	2	6	UPDATE (3, 5, "e", "E")	Undo
10	3	8	COMMIT	
11	3	10	END_TRANSACTION	
12			END_CHECKPOINT	
13	1	4	UPDATE (3, 6, "f", "F")	Undo
14	1	13	UPDATE (2, 7, "g", "G")	Undo

The recovery process starts at LSN 7 with the following transaction table and dirty pages table.

Transaction table:

TID	LSN
1	4
2	6
3	3

Dirty page table:

PID	LSN
1	4
2	5

Select the action taken during the undo passes for the UPDATE actions in the drop-down menus in the log above.