Discussion 12

Logging and Recovery

EECS 484

Logistics

- Final Exam on Tuesday, Dec. 13th, 7-9 PM EST
  - Exam location posted on Piazza
  - Please bring a **No. 2 pencil** and an **eraser**
  - One **double-sided, hand-written note sheet** (8.5"x11" letter size) and a **calculator** which must not have any wireless communication capability (Bluetooth, Wifi, etc)
- Today
  - Recovery
  - W21 practice exam

Recovery

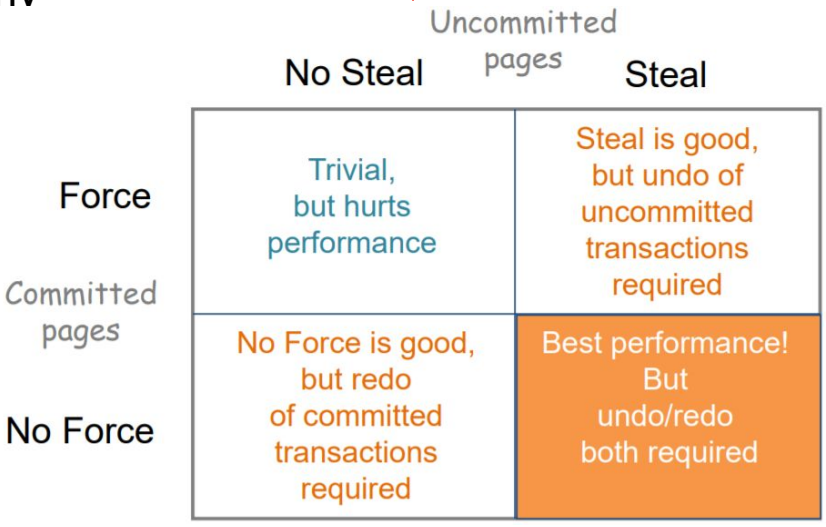
Recovery

- We’ve talked about the ACID properties
  - Need to actually enforce
- Potential problems
  - Committed transaction results not maintained
    - Example: Write result in memory, result does not get stored on disk
    - Power goes off, results lost :(
  - Uncommitted transaction results maintained
    - Example: Temporary results in memory get written to disk (not enough memory space)
    - Power goes off, results maintained :(

Workarounds

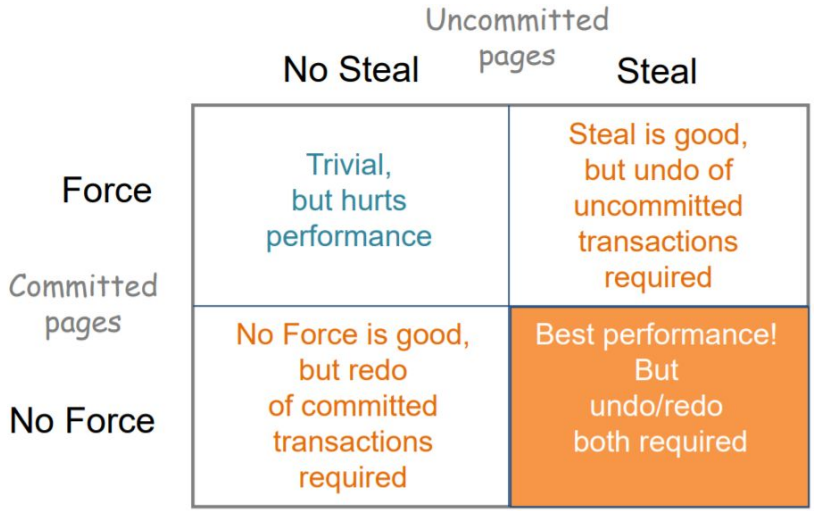
- We can workaround this by enforcing constraints on our transaction manager
  - FORCE pages of a transaction to disk upon a commit
    - Don't leave it in memory so once the commit happens, it's durable on the disk
    - Added IOs to potentially write back to memory early
  - NO STEAL pages of uncommitted transactions
    - Keep in memory so they don't get written earlv
    - Limited caching ability: can't evict some pag
- Constraints yield a valid database
  - Performance degrades significantly though

limit size of memory



ARIES

- Algorithm for Recovery and Isolation Exploiting Semantics (ARIES)
  - Allows us to perform Steal & No Force while maintaining ACID properties
  - Record the updates to memory and to a log
  - Use the log to check the system after a crash
    - Undo writes from uncommitted transactions that made it to disk
    - Redo writes from committed transactions that didn't make it to disk
  - Log must be stable storage (persist after crash)
    - Disk is stable storage as well
    - Memory is volatile (non-stable)



Log Entries

- Entry for each update we make to the database
- Contains information we need
  - Log Sequence Number (LSN)
  - Transaction ID (XID)
  - Previous Log Sequence Number (prevLSN) current transaction's previous actio
    - Backward pointer to last log entry for this transaction
  - Operation type:
    - Update, Commit, Abort, End (for Commit or Abort)
    - Compensation Log Record (Undo)
  - Page ID, Offset, Size, Old Data, New Data
    - Information about the data itself

Compensation Log Records (CLRs)

- Needed for undoing actions (after an abort or during Undo phase)
- Describes the undo action about to be performed
  - Remember we log before we do the action
  - Record of what the database looked like before the action we're undoing
    - The desired state after undo
  - We don't need to remember the state before the undo
    - We never undo an undo
  - Points to the next action to undo
    - Quick way to jump to the next action to undo

Cascading chain of CLRs pictured

Write Ahead Logging (WAL)

- Write Ahead Logging means we log before we write
- Force the log record for an update before the corresponding data page is written to the disk
  - Guarantees atomicity - we can undo stolen pages of transactions that were aborted
- Write all log records for a transaction before commits
  - Guarantees durability - we can redo committed pages that weren't forced

Transaction table and dirty page table

- Transaction table
  - Contains an entry for every currently active transaction
  - Each entry will have a transaction ID, status, and the LSN of the latest log record for this transaction (lastLSN)
- Dirty page table
  - Contains an entry for every page that has changes that have not yet been written back to disk
  - Each entry will have a page ID and the LSN of the first log record that caused the page to be dirty (recLSN)
  - Updates from which might have to be redone

Transaction table

tID	status	lastLSN
T1	active	
T2	active	

Dirty page table

pageID	recLSN
P484	
P370	

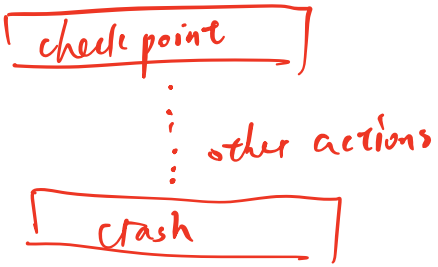
tID	type	pageID
T1	update	P484
T2	update	P370
T1	update	P484

Log

Checkpointing

- Every so often, the DBMS will save a copy of the transaction table and the dirty page table
  - DBMS will first create a “begin_checkpoint” record and add it to the log
  - DBMS will then create an “end_checkpoint” record that contains copies of the transaction and dirty page tables that are accurate *as of the time of the begin_checkpoint record* and add it to the log
  - Store the LSN of “begin_checkpoint” in the **master** record
- Allows the DBMS to start the analysis phase from the begin_checkpoint record rather than the very beginning of the log

Analysis Phase



- Recovery has three steps: Analysis, Redo, Undo
- Analysis reconstructs transaction and dirty-page tables at time of crash
  - Start at most recent checkpoint (begin_checkpoint)
  - Add all unfinished transactions to the transaction table (correct)
  - Add all modified pages to the dirty page table (≥ correct)
    - There may be more pages in the dirty page table than were actually dirty at the time
    - We do not log page flushes which means some pages may have been written to disk before the crash
  - No edits are made to the log
- Will use dirty-page table for redo and transaction table for undo phase

Redo Phase

- Recovery has three steps: Analysis, Redo, Undo
- Redo makes sure that all updates made before the crash are redone
  - Does not care if transaction committed before the crash or not
  - Start with the smallest **recLSN** from the dirty page table
  - For each log record (update/CLR) with **LSN**, check
    1. If the affected page, say P, is in the dirty page table
    2. If **LSN** >= **recLSN** of P
    3. If the **pageLSN** of P (on disk) < **LSN**
  - If all checks passed
    - Redo the update; set **pageLSN** of P = **LSN**
  - If any of the criteria are not met, then this action does not need to be reapplied
  - No edits are made to the log

Update:
check if it's in DPT
current LSN
If <: effect already on disk
latest update on that page

Undo Phase

- Recovery has three steps: Analysis, Redo, Undo
- Start from end of log and work backwards
  - Construct a set **ToUndo** of LSNs that contains the maximum LSN of each uncommitted transaction
    - Can retrieve these from the transaction table
    - Take the log entry corresponding to the maximum LSN from **ToUndo**
      - If it refers to an update, undo the update and add a CLR that points to the prevLSN
        - We chain our CLRs this way!
        - Add the prevLSN to **ToUndo**
      - If it's a CLR, just add the corresponding **undoNextLSN** to **ToUndo**
        - If undoNextLSN is NULL, add an end record to the log.
        - We never undo undos!
    - Remove the currently considered LSN from the set and continue until set is empty
- Only phase that adds to the log (in the form of CLRs)

add 'end Transaction' to log after undo

search log to find it

Recovery Example Question

LSN	LOG
1	update: T1 writes P102
2	update: T2 writes P102
3	update: T2 writes P103
4	begin_checkpoint
5	update: T1 writes P101
6	commit: T1
7	end_checkpoint
8	end: T1
9	abort: T2
10	CLR: Undo T2 LSN 3; undoNextLSN = ?
11	update: T3 writes P104
12	update: T3 writes P105
13	update: T4 writes P104
14	update: T5 writes P101
15	commit: T4
16	end: T4
CRASH, RESTART (Not a log record)	

Besides, a table showing the page information on the disk is found.

PageID	pageLSN
P101	5
P102	NULL
P103	NULL
P104	NULL
P105	12

LSN	LOG
1	update: T1 writes P102
2	update: T2 writes P102
3	update: T2 writes P103
4	begin_checkpoint
5	update: T1 writes P101
6	commit: T1
7	end_checkpoint
8	end: T1
9	abort: T2
10	CLR: Undo T2 LSN 3; undoNextLSN = ?
11	update: T3 writes P104
12	update: T3 writes P105
13	update: T4 writes P104
14	update: T5 writes P101
15	commit: T4
16	end: T4
CRASH, RESTART (Not a log record)	

Q: What is the undoNextLSN value of the log record with LSN = 10?

A: **2**

LSN	LOG
1	update: T1 writes P102
2	update: T2 writes P102
3	update: T2 writes P103
4	begin_checkpoint
5	update: T1 writes P101
6	commit: T1
7	end_checkpoint
8	end: T1
9	abort: T2
10	CLR: Undo T2 LSN 3; undoNextLSN = ?
11	update: T3 writes P104
12	update: T3 writes P105
13	update: T4 writes P104
14	update: T5 writes P101
15	commit: T4
16	end: T4
CRASH, RESTART (Not a log record)	

Besides, a table showing the page information on the disk is found.

PageID	pageLSN
P101	5
P102	NULL
P103	NULL
P104	NULL
P105	12

Q: Give the Transaction Table and Dirty Page Table stored within the checkpoint. (***For DPT, assume no flushes before begin_checkpoint.)***

A:

LSN	LOG
1	update: T1 writes P102
2	update: T2 writes P102
3	update: T2 writes P103
4	begin_checkpoint
5	update: T1 writes P101
6	commit: T1
7	end_checkpoint
8	end: T1
9	abort: T2
10	CLR: Undo T2 LSN 3; undoNextLSN = ?
11	update: T3 writes P104
12	update: T3 writes P105
13	update: T4 writes P104
14	update: T5 writes P101
15	commit: T4
16	end: T4
CRASH, RESTART (Not a log record)	

Besides, a table showing the page information on the disk is found.

PageID	pageLSN
P101	5
P102	NULL
P103	NULL
P104	NULL
P105	12

Q: Give the Transaction Table and Dirty Page Table stored within the checkpoint. (***For DPT, assume no flushes before begin_checkpoint.)***

A:

Transaction Table	
TxID	lastLSN
T1	1
T2	3

Dirty Page Table	
PgID	recLSN
P102	1
P103	3

LSN	LOG
1	update: T1 writes P102
2	update: T2 writes P102
3	update: T2 writes P103
4	begin_checkpoint
5	update: T1 writes P101
6	commit: T1
7	end_checkpoint
8	end: T1
9	abort: T2
10	CLR: Undo T2 LSN 3; undoNextLSN = ?
11	update: T3 writes P104
12	update: T3 writes P105
13	update: T4 writes P104
14	update: T5 writes P101
15	commit: T4
16	end: T4
CRASH, RESTART (Not a log record)	

Q: Which LSN is stored in the master record?

A: **4**

LSN	LOG
1	update: T1 writes P102
2	update: T2 writes P102
3	update: T2 writes P103
4	begin_checkpoint
5	update: T1 writes P101
6	commit: T1
7	end_checkpoint
8	end: T1
9	abort: T2
10	CLR: Undo T2 LSN 3; undoNextLSN = ?
11	update: T3 writes P104
12	update: T3 writes P105
13	update: T4 writes P104
14	update: T5 writes P101
15	commit: T4
16	end: T4
CRASH, RESTART (Not a log record)	

Q: Which LSN should the analysis phase start from?

A: **4** master record

LSN	LOG
1	update: T1 writes P102
2	update: T2 writes P102
3	update: T2 writes P103
4	begin_checkpoint
5	update: T1 writes P101
6	commit: T1
7	end_checkpoint
8	end: T1
9	abort: T2
10	CLR: Undo T2 LSN 3; undoNextLSN = ?
11	update: T3 writes P104
12	update: T3 writes P105
13	update: T4 writes P104
14	update: T5 writes P101
15	commit: T4
16	end: T4
CRASH, RESTART (Not a log record)	

Transaction table and dirty page table stored within the checkpoint:

Transaction Table	
TxID	lastLSN
T1	1
T2	3

Dirty Page Table	
PgID	recLSN
P102	1
P103	3

Q: Give the Transaction Table and Dirty Page Table after the analysis phase is done.

A:

LSN	LOG
1	update: T1 writes P102
2	update: T2 writes P102
3	update: T2 writes P103
4	begin_checkpoint
5	update: T1 writes P101
6	commit: T1
7	end_checkpoint
8	end: T1
9	abort: T2
10	CLR: Undo T2 LSN 3; undoNextLSN = ?
11	update: T3 writes P104
12	update: T3 writes P105
13	update: T4 writes P104
14	update: T5 writes P101
15	commit: T4
16	end: T4
CRASH, RESTART (Not a log record)	

Transaction table and dirty page table stored within the checkpoint:

Transaction Table	
TxID	lastLSN
T1	1
T2	3

Dirty Page Table	
PgID	recLSN
P102	1
P103	3

Q: Give the Transaction Table and Dirty Page Table after the analysis phase is done.

A:

Transaction Table	
TxID	lastLSN
T2	10
T3	12
T5	14

Dirty Page Table	
PgID	recLSN
P101	5
P102	1
P103	3
P104	11
P105	12

LSN	LOG
1	update: T1 writes P102
2	update: T2 writes P102
3	update: T2 writes P103
4	begin_checkpoint
5	update: T1 writes P101
6	commit: T1
7	end_checkpoint
8	end: T1
9	abort: T2
10	CLR: Undo T2 LSN 3; undoNextLSN = ?
11	update: T3 writes P104
12	update: T3 writes P105
13	update: T4 writes P104
14	update: T5 writes P101
15	commit: T4
16	end: T4
CRASH, RESTART (Not a log record)	

Transaction table and dirty page table after analysis phase:

Transaction Table	
TxID	lastLSN
T2	10
T3	12
T5	14

Dirty Page Table	
PgID	recLSN
P101	5
P102	1
P103	3
P104	11
P105	12

smallest

Q: Which LSN should the redo phase start from?

A: **1**

LSN	LOG
1	update: T1 writes P102
2	update: T2 writes P102
3	update: T2 writes P103
4	begin_checkpoint
5	update: T1 writes P101
6	commit: T1
7	end_checkpoint
8	end: T1
9	abort: T2
10	CLR: Undo T2 LSN 3; undoNextLSN = ?
11	update: T3 writes P104
12	update: T3 writes P105
13	update: T4 writes P104
14	update: T5 writes P101
15	commit: T4
16	end: T4
CRASH, RESTART (Not a log record)	

Transaction table and dirty page table after analysis phase:

Transaction Table	
TxID	lastLSN
T2	10
T3	12
T5	14

Dirty Page Table	
PgID	recLSN
P101	5
P102	1
P103	3
P104	11
P105	12

Q: Which operations must be redone? Give the corresponding LSNs.

A: 1 2 3 10 11 13 14

PageID	pageLSN
P101	5
P102	NULL
P103	NULL
P104	NULL
P105	12

LSN	LOG
1	update: T1 writes P102
2	update: T2 writes P102
3	update: T2 writes P103
4	begin_checkpoint
5	update: T1 writes P101
6	commit: T1
7	end_checkpoint
8	end: T1
9	abort: T2
10	CLR: Undo T2 LSN 3; undoNextLSN = ?
11	update: T3 writes P104
12	update: T3 writes P105
13	update: T4 writes P104
14	update: T5 writes P101
15	commit: T4
16	end: T4
CRASH, RESTART (Not a log record)	

Transaction table and dirty page table after analysis phase:

Transaction Table	
TxID	lastLSN
T2	10
T3	12
T5	14

Dirty Page Table	
PgID	recLSN
P101	5
P102	1
P103	3
P104	11
P105	12

Q: Which LSN should the undo phase start from?

A: 14

LSN	LOG
1	update: T1 writes P102
2	update: T2 writes P102
3	update: T2 writes P103
4	begin_checkpoint
5	update: T1 writes P101
6	commit: T1
7	end_checkpoint
8	end: T1
9	abort: T2
10	CLR: Undo T2 LSN 3; undoNextLSN = ~~1~~ 2
11	update: T3 writes P104
12	update: T3 writes P105
13	update: T4 writes P104
14	update: T5 writes P101
15	commit: T4
16	end: T4
CRASH, RESTART (Not a log record)	

Transaction table and dirty page table after analysis phase:

Transaction Table	
TxID	lastLSN
T2	10
T3	12
T5	14

Dirty Page Table	
PgID	recLSN
P101	5
P102	1
P103	3
P104	11
P105	12

Q: Complete the LOG table after the recover process.

A: in the next page

{10, 12, 14}

① undo 14
② no prev on T5
③ Add end T5 to log
① Pick 12 (undo 12)
② add CLR and undonext = 11
add 11

LSN	LOG
1	update: T1 writes P102
2	update: T2 writes P102
3	update: T2 writes P103
4	begin_checkpoint
5	update: T1 writes P101
6	commit: T1
7	end_checkpoint
8	end: T1
9	abort: T2
10	CLR: Undo T2 LSN 3; undoNextLSN = ?
11	update: T3 writes P104
12	update: T3 writes P105
13	update: T4 writes P104
14	update: T5 writes P101
15	commit: T4
16	end: T4
CRASH, RESTART	
17	CLR: Undo T5 LSN 14; undoNextLSN = NULL
18	end: T5
19	CLR: Undo T3 LSN 12; undoNextLSN = 11
20	CLR: Undo T3 LSN 11; undoNextLSN = NULL
21	end: T3
22	CLR: Undo T2 LSN 2; undoNextLSN = NULL
23	end: T2

③ Pick 11 (undo 11) undonext = ∅

④ Add end T3 to log

① Pick 10 (CLR)

② add 2

③ undo2 add CLR