COMP 163

Lecture 21

RECOVERY: ARIES

ARIES

- ARIES
  - Algorithm for Recovery and Isolation Exploiting Semantics

- ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging
  - Mohan, Hederle, Lindsey, Pirahesh, Schwartz (IBM)
  - ACM TODS, 1992

Source

- This lecture follows the presentation in *Database Management Systems, 3rd Ed.*
  - Ramakrishnan and Gehrke
  - McGraw Hill, 2003

Crash Recovery: IBM DB2, Informix, Microsoft SQL Server, Oracle 8, and Sybase ASE all use a WAL scheme for recovery. IBM DB2 uses ARIES, and the others use schemes that are actually quite similar to ARIES (e.g., all changes are re-applied, not just the changes made by transactions that are 'winners') although there are several variations.

Overview - ARIES

- recovery algorithm designed for steal/no-force buffer management
  - steal = buffer pages not pinned by active transactions
  - no-force = buffer pages not forced to disk at transaction commit
  - *buffer = cache = main memory copies of pages*

Overview – ARIES

- After a crash, three phase restart:
  - ANALYSIS: Identify dirty pages in buffer and active transactions
  - REDO: repeat all actions to bring DB back to state at time of crash
  - UNDO: rollback active transaction

ARIES Principles

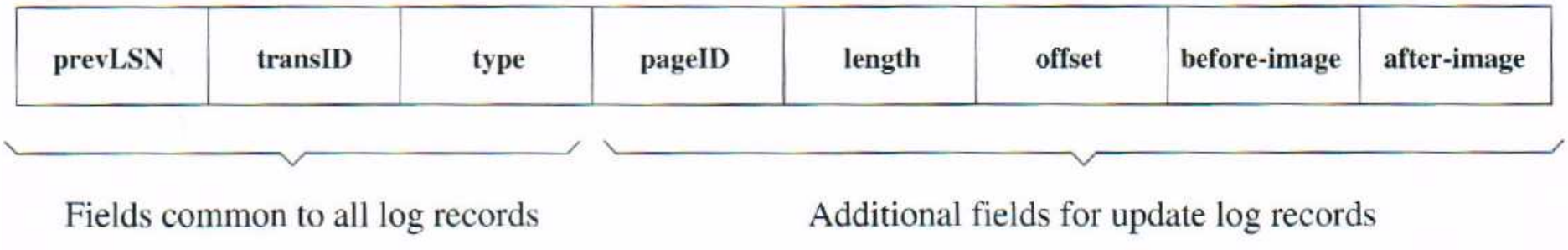
- Write-Ahead Logging (WAL)
  - all changes must be logged to stable storage before data is changed on disk
- Repeating History (REDO)
  - get DB back to state that existed at time of crash
  - allows for fine granularity locks
- Logging changes during UNDO
  - changes made during UNDO are logged
  - prevents repetition in case of repeated restarts (due to crash during recovery)

The Log

- Must be on “stable storage”
- Log records have unique IDs
  - log sequence numbers (LSN)
  - must be monotonic in time order
- pageLSN
  - each page of database tagged with most recent log record affect that page

Log Records

- Log Records added when
  - Updating a Page (also updates pageLSN)
  - Transaction Commit or Abort
  - Transaction End
    - after commit/abort actually finish
  - UNDO operation
    - compensating log record (CLR)



Logging commit/abort/end

- Commit
  - commit record added to log
  - log is force written to storage
  - transaction removed from transaction table
  - (data pages not force written)
- Abort
  - abort record added to log
  - UNDO run for transaction
- End
  - appended to log after above action complete

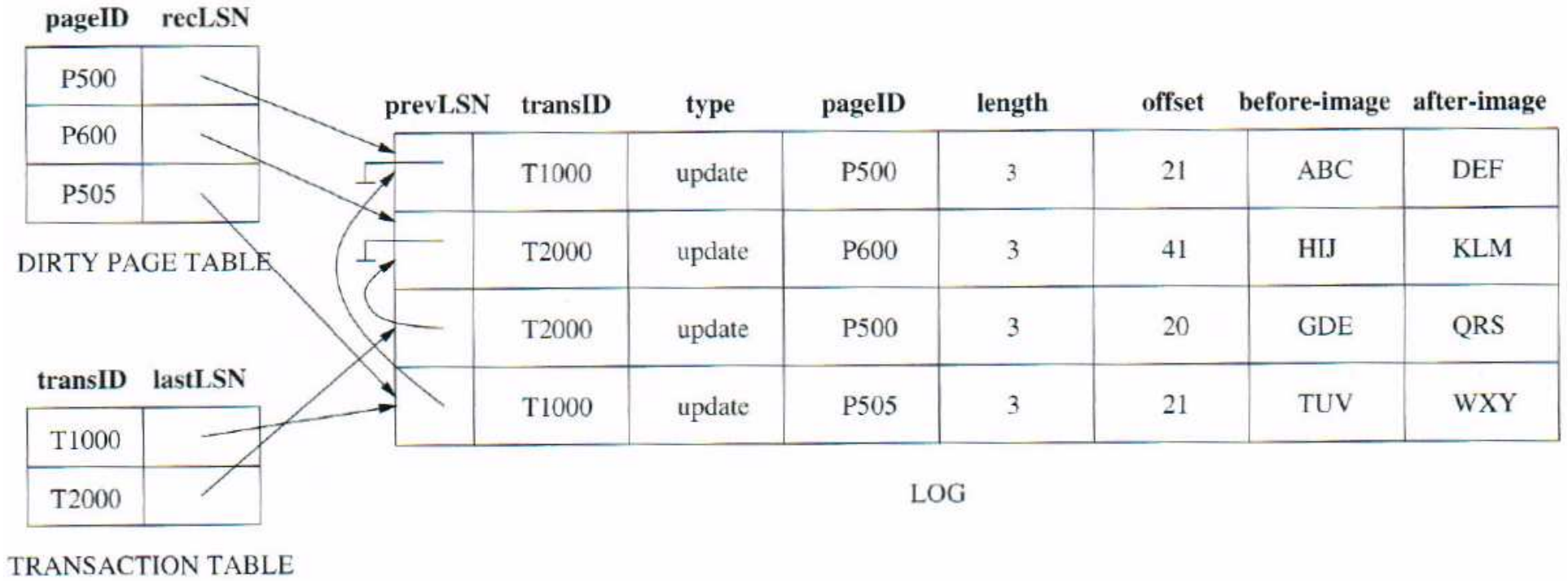
Compensation Log Records

- A CLR is logged before an update record is undone
- CLR is similar to an update record, but describes the change for the undo
- CLR also contains undoNextLSN
  - LSN of next update that will need to be undone
- CLR actions never need to be undone
  - may be redone, in case of crash during recovery

Other Recovery Tables

- Transaction Table (TT)
  - for each active transaction:
    - transaction id
    - lastLSN (most recent log record)
    - status (active, committed or aborted)
  - entry removed when transaction reached END
- Dirty Page Table (DPT)
  - for each dirty buffer page, stores recLSN (earliest log record that might need to be redone for page)

Example Recovery Tables



pageID	recLSN
P500	
P600	
P505	

DIRTY PAGE TABLE

transID	lastLSN
T1000	
T2000	

TRANSACTION TABLE

prevLSN	transID	type	pageID	length	offset	before-image	after-image
	T1000	update	P500	3	21	ABC	DEF
	T2000	update	P600	3	41	HIJ	KLM
	T2000	update	P500	3	20	GDE	QRS
	T1000	update	P505	3	21	TUV	WXY

LOG

WAL

- WAL is the fundamental rule that ensures that a record of every change to DB is available during recovery
- Without WAL, no way to ensure durability
- Allows definition of “committed transaction”:
  - *a transaction all of whose log records, including a commit record, have been written to stable storage*
- since Log is sequential, update cost is minimal (one block write)

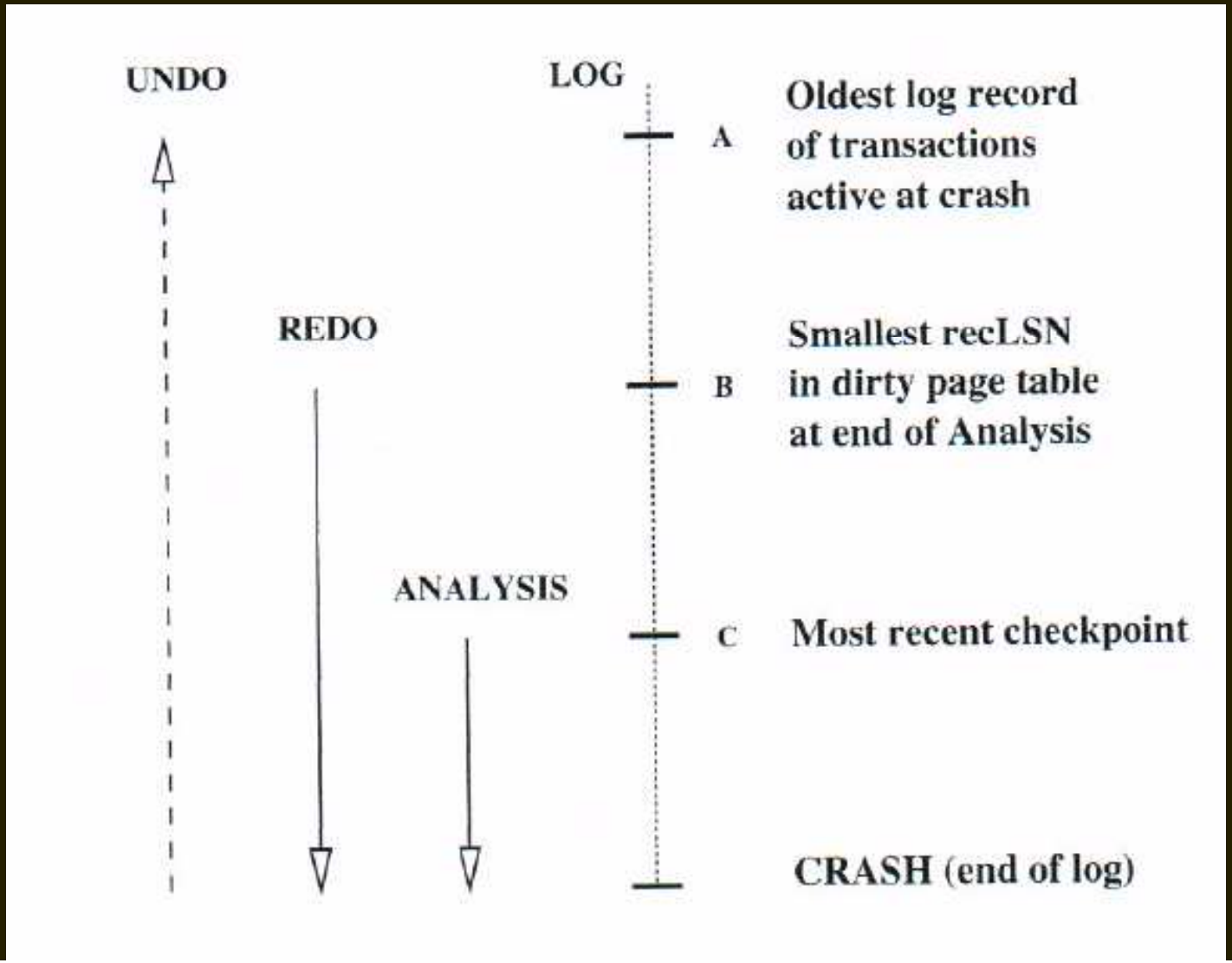
Checkpoints

- checkpoints can reduce work needed during recovery
- ARIES uses fuzzy checkpoint
  - does not require idle DB or buffer flush
- guarantees we have log, transaction table and dirty page table as of time of checkpoint start

Checkpoint Process

- log begin_checkpoint
- construct end_checkpoint
  - copy TT and DPT
- log end_checkpoint
- write special master record
  - LSN of begin_checkpoint
  - written to reserved space in stable storage

Recovery Phases



Analysis Phase Tasks

- Determine point in log to begin REDO
- Determine (conservative) set of dirty pages at time of crash
- Identify active transactions at time of crash

Analysis Phase Process

- TT and DPT restored to state at last checkpoint
- log is scanned forward from checkpoint
  - see an end record for xact T, remove T from transaction table
  - see any other record for xact T, and T is not in TT, add T to TT
  - If a redoable record for page P is seen, and P is not in DPT, add P to DPT
- TT and DPT now restored to state at time of crash

REDO Process

- Begin at log record indicated by smallest recLSN in DPT
- Reapplies, in order, updates from all xacts
- If a xact was aborted and CLRs are in log, reapply the CLRs

- Result: DB, TT and DPT all back to state at time of crash

UNDO process

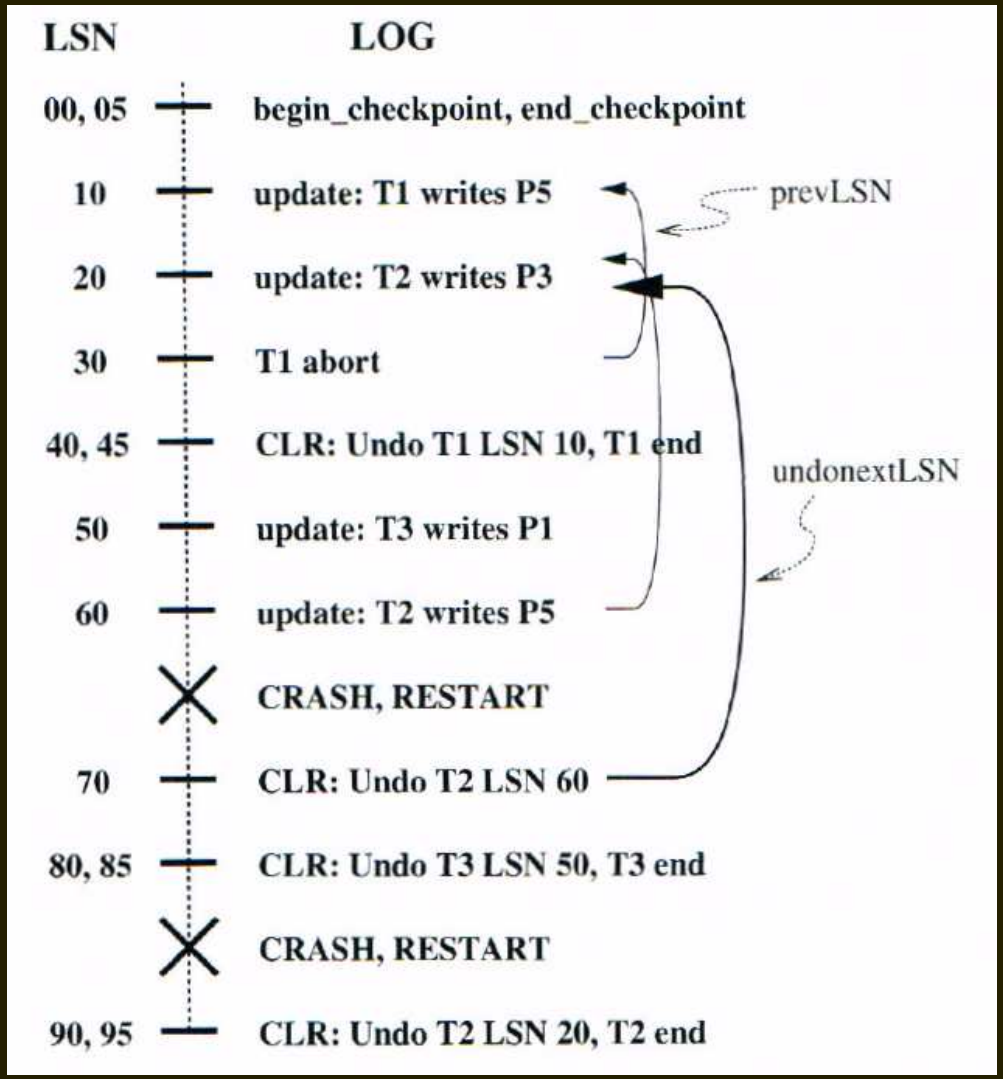
- Check TT for set of active transactions
- Process log in reverse order, until all active transactions are undone
- log CLRs as update records are undone

- Result: DB is now in a consistent state
  - effects of committed xacts have been restored
  - effects of uncommitted xacts have been removed

Transaction Abort/Rollback

- Aborting a transaction during normal operation is now a special case of the ARIES UNDO phase
- Simply start with a set of one xact to undo

Crashes During Recovery



Dirty Page Table

pageID	recLSN

Transaction Table

transID	lastLSN

Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	

OPERATIONS:
begin_checkpoint
end_checkpoint

Note: we'll ignore the actual values read/written and omit the BFIM/AFIM

Dirty Page Table

pageID	recLSN
5	10

Transaction Table

transID	lastLSN
1	10

Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5

OPERATIONS:
$w_1(P5)$

Dirty Page Table

pageID	recLSN
5	10
3	**20**

Transaction Table

transID	lastLSN
1	10
2	**20**

Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		**2**	**update**	**3**

OPERATIONS:
$w_2(P3)$

Dirty Page Table

pageID	recLSN
5	10
3	20

Transaction Table

transID	lastLSN
1	10
2	20

Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		2	update	3
30		1	abort	

OPERATIONS:
a_1 (abort)

Now we need to undo xact 1.
From TT, start at LSN 10. Restore BFIM in Page 5 (page 5 is still dirty).
Add CLR to log and end xact.

Dirty Page Table

pageID	recLSN
5	10
3	20

Transaction Table

transID	lastLSN
~~1~~	~~45~~
2	20

Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		2	update	3
30	10	1	abort	
40	**30 (Ø)**	**1**	**CLR 10**	**5**
45	**40**	**1**	**end**	

OPERATIONS:
a_1 (abort)

The value in parentheses in prevLSN column is the undoNextLSN, which indicates the next record that will need to be undone for the current xact.

Dirty Page Table

pageID	recLSN
5	10
3	20
1	**50**

Transaction Table

transID	lastLSN
2	20
3	**50**

Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		2	update	3
30	10	1	abort	
40	30 (Ø)	1	CLR 10	5
45	40	1	end	
50		**3**	**update**	**1**

OPERATIONS:
$w_3(P1)$

Dirty Page Table

pageID	recLSN
5	10
3	20
1	50

Transaction Table

transID	lastLSN
2	**60**
3	50

Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		2	update	3
30	10	1	abort	
40	30 (Ø)	1	CLR 10	5
45	40	1	end	
50		3	update	1
60	**20**	**2**	**update**	**5**

OPERATIONS:
$w_2(P5)$

Dirty Page Table

pageID	recLSN

Transaction Table

transID	lastLSN

Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		2	update	3
30	10	1	abort	
40	30 (Ø)	1	CLR 10	5
45	40	1	end	
50		3	update	1
60	20	2	update	5

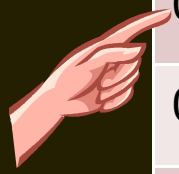
CRASH!

Dirty Page Table

pageID	recLSN

Transaction Table

transID	lastLSN



Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		2	update	3
30	10	1	abort	
40	30 (Ø)	1	CLR 10	5
45	40	1	end	
50		3	update	1
60	20	2	update	5

Analysis restores TT & DPT
and starts at last checkpoint

Dirty Page Table

pageID	recLSN
5	10

Transaction Table

transID	lastLSN
1	10

Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		2	update	3
30	10	1	abort	
40	30 (Ø)	1	CLR 10	5
45	40	1	end	
50		3	update	1
60	20	2	update	5

Analysis

Dirty Page Table

pageID	recLSN
5	10
3	20

Transaction Table

transID	lastLSN
1	10
2	20

Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		2	update	3
30	10	1	abort	
40	30 (Ø)	1	CLR 10	5
45	40	1	end	
50		3	update	1
60	20	2	update	5

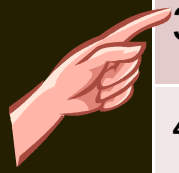
Analysis

Dirty Page Table

pageID	recLSN
5	10
3	20

Transaction Table

transID	lastLSN
1	30
2	20



Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		2	update	3
30	10	1	abort	
40	30 (Ø)	1	CLR 10	5
45	40	1	end	
50		3	update	1
60	20	2	update	5

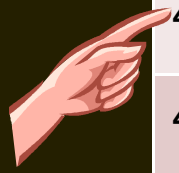
Analysis

Dirty Page Table

pageID	recLSN
5	10
3	20

Transaction Table

transID	lastLSN
1	40
2	20



Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		2	update	3
30	10	1	abort	
40	30 (Ø)	1	CLR 10	5
45	40	1	end	
50		3	update	1
60	20	2	update	5

Analysis

Dirty Page Table

pageID	recLSN
5	10
3	20

Transaction Table

transID	lastLSN
2	20

Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		2	update	3
30	10	1	abort	
40	30 (Ø)	1	CLR 10	5
45	40	1	end	
50		3	update	1
60	20	2	update	5

Analysis

Dirty Page Table

pageID	recLSN
5	10
3	20
1	50

Transaction Table

transID	lastLSN
2	20
3	50

Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		2	update	3
30	10	1	abort	
40	30 (Ø)	1	CLR 10	5
45	40	1	end	
50		3	update	1
60	20	2	update	5

Analysis

Dirty Page Table

pageID	recLSN
5	10
3	20
1	50

Transaction Table

transID	lastLSN
2	60
3	50

Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		2	update	3
30	10	1	abort	
40	30 (Ø)	1	CLR 10	5
45	40	1	end	
50		3	update	1
60	20	2	update	5

Analysis - TT and DPT restored

Dirty Page Table

pageID	recLSN
5	10
3	20
1	50

Transaction Table

transID	lastLSN
2	60
3	50

Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		2	update	3
30	10	1	abort	
40	30 (Ø)	1	CLR 10	5
45	40	1	end	
50		3	update	1
60	20	2	update	5

REDO – apply updates to buffer pages (not shown)

Dirty Page Table

pageID	recLSN
5	10
3	20
1	50

Transaction Table

transID	lastLSN
2	60
3	50

Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		2	update	3
30	10	1	abort	
40	30 (Ø)	1	CLR 10	5
45	40	1	end	
50		3	update	1
60	20	2	update	5

UNDO – active xacts = { 2, 3 }
undo in reverse order

Dirty Page Table

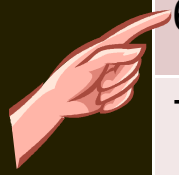
pageID	recLSN
5	10
3	20
1	50

Transaction Table

transID	lastLSN
2	70
3	50

Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		2	update	3
30	10	1	abort	
40	30 (Ø)	1	CLR 10	5
45	40	1	end	
50		3	update	1
60	20	2	update	5
70	60 (20)	2	CLR 60	5



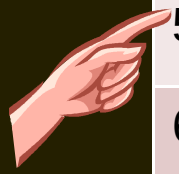
UNDO – xact 2 is not finished (by prevLSN)

Dirty Page Table

pageID	recLSN
5	70
3	20
1	50

Transaction Table

transID	lastLSN
2	70
~~3~~	~~80~~



Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		2	update	3
30	10	1	abort	
40	30 (Ø)	1	CLR 10	5
45	40	1	end	
50		3	update	1
60	20	2	update	5
70	60 (20)	2	CLR 60	5
80	50 (Ø)	3	CLR 50	1
85	80	3	end	

UNDO – xact 3 is finished
(by NULL prevLSN)

Dirty Page Table

pageID	recLSN

Transaction Table

transID	lastLSN

CRASH!

Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		2	update	3
30	10	1	abort	
40	30 (Ø)	1	CLR 10	5
45	40	1	end	
50		3	update	1
60	20	2	update	5
70	60 (20)	2	CLR 30	5
80	50 (Ø)	3	CLR 50	1
85	80	3	end	

Dirty Page Table

pageID	recLSN
5	70
3	20
1	50

Transaction Table

transID	lastLSN
2	70

Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		2	update	3
30	10	1	abort	
40	30 (Ø)	1	CLR 10	5
45	40	1	end	
50		3	update	1
60	20	2	update	5
70	60 (20)	2	CLR 30	5
80	50 (Ø)	3	CLR 50	1
85	80	3	end	

ANALYSIS rebuilds
TT and DPT (omitted)
REDO updates buffer pages

UNDO finds active xact = { 2 }

Dirty Page Table

pageID	recLSN
5	10
3	20
1	50

Transaction Table

transID	lastLSN
~~2~~	~~70~~

Log

LSN	prevLSN	transID	type	pageID
00			beginCP	
05			endCP	
10		1	update	5
20		2	update	3
30	10	1	abort	
40	30 (Ø)	1	CLR 10	5
45	40	1	end	
50		3	update	1
60	20	2	update	5
70	60 (20)	2	CLR 30	5
80	50 (Ø)	3	CLR 50	1
85	80	3	end	
90	**20 (Ø)**	**2**	**CLR 20**	**3**
95	**90**	**2**	**end**	

UNDO starts at record 20 (based on undoNextLSN at record 70)

xact 2 completely undone