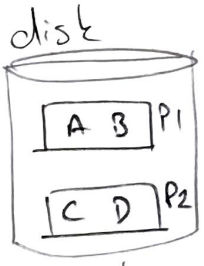
week9

ex:

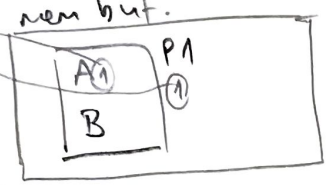


T1 w A
T2 w B
T2 w C
Sys flush log to disk + P2 to disk
T1 w D
T1 commit. System writes a com. log and flushes log to disk.
T2 w B
T1 END → to log (in memory)
Crash

---

a) Yukarıdaki stepleri canlandıralım. Init mem buf, txn, dp } all 0.

1) $W_1(A)$



dp

pid	rec lsn
~~P1~~	1

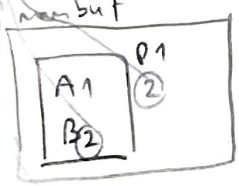
txn

xid	last lsn	state
T1	1	run

Log (in memory)

lsn	xid	prevLsn	type	pid	log	undo next lsn
1	T1	–	update	P1	W(A→A1)	–

2) $W_2(B)$



dp

P1	1

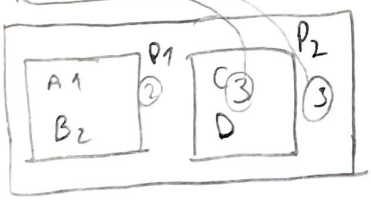
txn

T1	1	run
T2	2	run

Log | mem

1	–	–	–			
2	T2	–	upd	P1	W(B→B2)	–

3) $W_2(C)$



dp

P1	1
P3	3

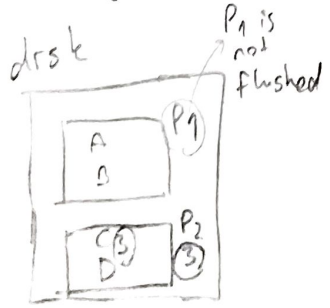
txn

T1	1	r
T2	3	r

Log | mem

1	–	–	–			
2						
3	T2	2	upd	P2	W(C→C3)	–

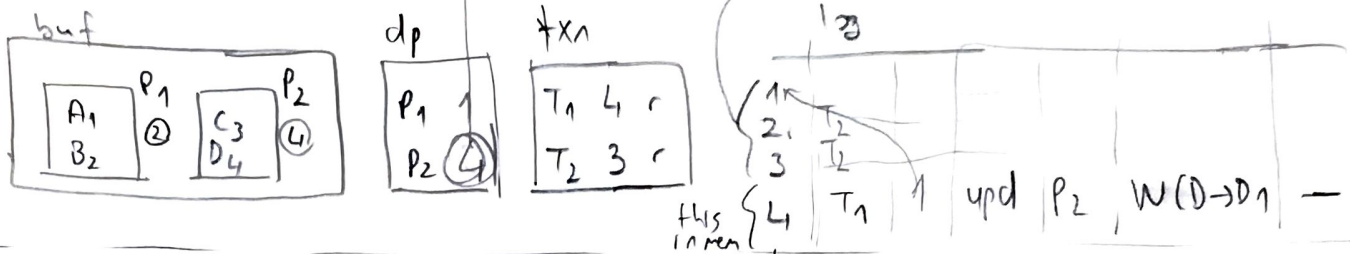
4) Now system flushes to disk (log and P_2)



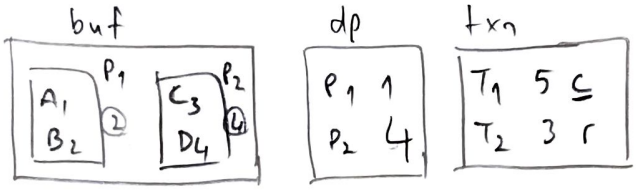
disk log

1	
2	
3	–

4) $W_1(D)$ — Since P_2 is flushed it was not dirty, update → this is in disk



5) T_1 commits, sys writes commit to log to disk

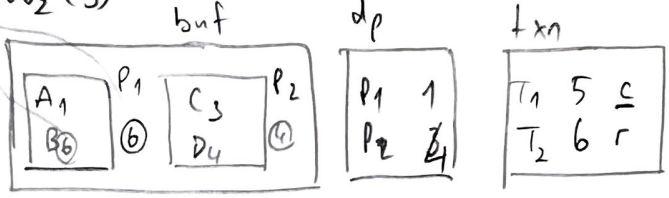


Log in disk

1
2
3
4 T_1 1
5 T_1 4 com. — — —

} all in the disk

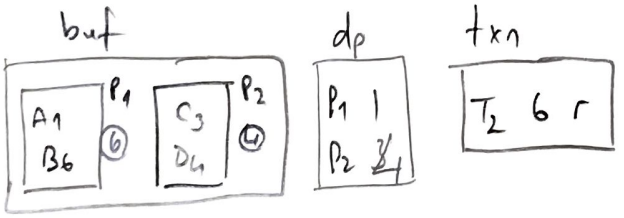
6) $W_2(B)$



Log in mem

6 T_2 3 upd P_1 $W(B \to B_6)$ —

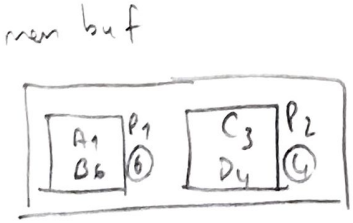
7) T_1 end → write that to log
delete T_1 from active txn



Log in mem

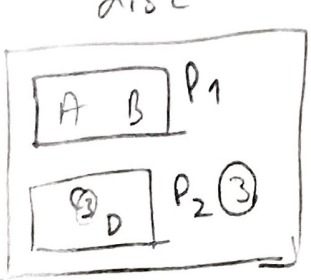
6 —
7 T_1 5 end — — —

CRASH So at that time we have



Log in mem

1
2
3
4
5
6
7 T_1 end — —



Log in disk ☆

1
2
3
4
✱5 T_1 4 com.

b) Analysis Phase

Look at the log, start from checkpoint. Build dp & active txn.

Log in disk

1	T_1	–	up	P_1	$W(A \to A_1)$	—
2	T_2	–	up	P_1	$(B \to B_2)$	—
3	T_2	2	up	P_2	$(C \to C_3)$	—
4	T_1	1	up	P_2	$(D \to D_4)$	
5	T_1	4	C	–	—	—

now, dp, txn and buffer is empty

1) dp

P_1	1

active txn

T_1	1	r

2) dp

P_1	1

txn

T_1	1	r
T_2	2	r

3) dp

P_1	1
P_2	3

txn

T_1	1	r
T_2	3	r

4) dp

P_1	1
P_2	(3)

→ in the normal version, we have flushed and updated. However, the log does not know that

txn

T_1	4	r
T_2	3	r

5) dp

P_1	(1)
P_2	3

txn

T_1	5	com
T_2	3	r.

⇒ we have build it. Remember, we have

A B	P_1	C_3 D	P_2	(3)

in disk

c) Redo Phase

Start redoing the log from the lowest recLSN.
We didn't go back off 1, but we might. (Bu sınıfta checkpointten daha küçük recLsn olmadı. Olabilirdi işlemleri log'a bakarak yapıyorum.)

1) Bring P_1 to buffer. $W_1(A)$

A_1 B	P_1	(1)

2) $W_2(B)$

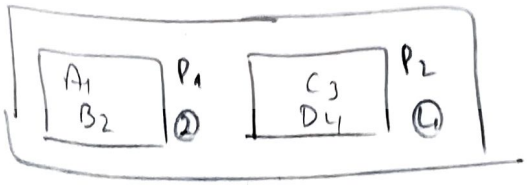
A_1 B_2	P_1	(2)

3) $W_2(C)$. Bring P_2 to buffer.

P_2'nin diskte numarası 3 bu işlem için de 3. Bu işlemi yapmıyorum.

A_1 B_2	P_1 (2)	C_3 D	P_2 (3)	

4) $W_1(D)$



During analysis and redo, if another crash happens, redo that steps again. ↑ don't write anything to log

5) end of log.

Now, we have

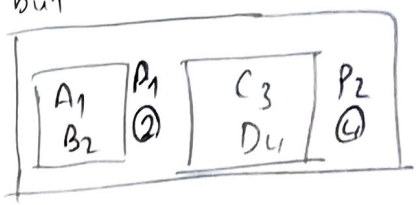
dp

P_1	1
P_2	3

txn

T_1	5	com
T_2	3	r

buf



d) Undo

It knows T_1 committed. It has to write end for T_1, and should abort T_2.

6) Log in mem

Lsn	trid	prevLsn	type	pid	log entry	undo nextLsn
6	T_1	5	end	—	—	—

delete T_1 from txn

~~T_1~~	~~5~~	~~com~~
T_2	3	abort

Now, it should undo T_2. look at log. T_2's last entry is ③.

Undo {3}

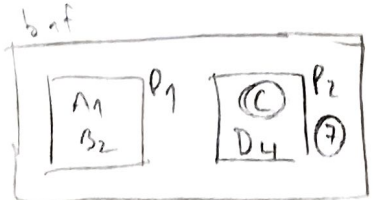
7) Write compensation log record (CLR) to log.

Log in mem

6	—					
7	T_2	3	CLR	—	undo T_2 Lsn 3	2

→ 3 to be undo Lsn 2 id.

buf



txn

T_2	7	abort

Now, Undo(3) is finished. But it has link to 2.

8) Undo(2)

2	T_2	-	Upd	P_1	$W(B \to B_2)$

log

6
7
8 T_2 - CLR Undo T_2 Lsn 2 → T_2 does not have an entry prev. than 2.

buf

A_1	P_1	C	P_2
B	8	D_4	7

txn

T_2	8	abort

9) Undo( ) is empty. nothing to do

Log

6
7
8
9 T_2 8 End - - -

Now, if you flush that log to disk

disk

A_1	P_1
B	8

C	P_2
D_4	7

If there is a crash before writing (flushing) log to disk. Redo again.

Checkpointing

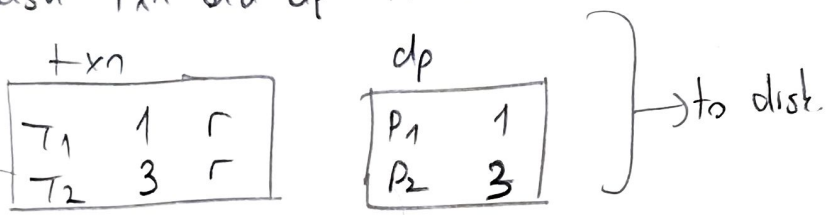
Suppose in the example

1 T_1 — —
2 T_2 — —
3 T_2 2 upd P_2 W(C→C_3)

checkpointing occured.

When checkpointing is occured;

Flush txn and dp to disk



also, write begin ckpt end // to log

Suppose a crash happened at 5.

Analysis should start from there.

1) Load txn and dp

2) Start analyse from 4th log. ⇒ Daha az analiz yapmış olduk.