Winter 2022 - CS540 – Assignment 6 (Written)

Full Name	OSU Email ID	ONID
Bharghav Srikhakollu	srikhakb@oregonstate.edu	934289891
Akhil Sai Chintala	chintala@oregonstate.edu	934409906

1. Recovery:

(a) Describe the steal and no-force policies.

Steal: When a transaction(T1) is running for some time and another transaction(T2) started with frames in the bufferpool and it needs more frames. Here we steal the frames from T1 and will give it to T2. At the time of giving frames to T2, we make sure that the latest contents of T1 are written to the disk. This phenomenon can be defined as steal policy. With steal policy we cannot guarantee atomicity but there is effective use of buffer (main memory).

In other words, we can say that changes made to an object in bufferpool by a transaction can be written to disk before the transaction commits. This happens because another transaction might steal buffer page which is currently occupied by an uncommitted transaction.

No-force: During running of a transaction, we do not force transaction to write to disk. We will let it do all changes in buffer (main memory) and will write to disk only when the buffer manager decides to write. Such policy can be termed as no-force policy. With no-force policy we cannot guarantee durability, but it is efficient in terms of response time.

In other words, when a transaction commits, we need not ensure that all changes made to the objects in the buffer pool are immediately forced to the disk. Such phenomena can be termed as no-force policy.

Reference citation: Class lecture and link below.
https://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/solutions/ans3ed-oddonly.pdf

2: Recovery and ARIES:

(a) Where in the log file does the ARIES algorithm start redoing log records in the redo phase of recovery after a crash?

According to the ARIES algorithm, we have Analysis phase, REDO phase and UNDO phase. During the redo phase of recover after a crash, the phase begins with the log record which has the “smallest recLSN” of all the pages in the dirty page table which is built during the analysis phase. This log record is chosen as it identifies the oldest update which may not be written to disk prior to the crash. From this point, REDO scans forward until the end of the log.

(b) Consider a system uses ARIES for logging and recovery. Assume that a transaction commits during the normal execution of the database system. What log records of this transaction will be on the log file?

When a transaction commits during the normal execution of the database system, it force-writes a “commit” type log record on the log file. This log record contains transaction id. The log record is appended to the log and the log tail is written to stable storage, up to and including the commit record.

Then update the transaction status to commit and will flush the dirty pages to disk. At the end we write end record to the log. In the final stage, we will update the transaction status as complete.

Reference citation: Cow book – 20.1.1 The log.

3: Recovery and ARIES:

In this problem, you need to simulate the actions taken by the ARIES algorithm. Consider the following log records and buffer actions:

Time	LSN	Log	Buffer actions
0	00	update: T1 updates P7	P7 brought into the buffer
1	10	update: T0 updates P9	P9 brought into the buffer; P9 flushed to disk
2	20	update: T1 updates P8	P8 brought into the buffer; P8 flushed to disk
3	30	begin checkpoint	
4	40	end checkpoint	
5	50	update: T1 updates P9	P9 brought into the buffer
6	60	update: T2 updates P6	P6 brought into the buffer
7	70	update: T1 updates P5	P5 brought into the buffer
8	80	update: T1 updates P7	P6 flushed to disk
9		CRASH RESTART	

(a) For the actions listed above, show Transaction Table (XT) and Dirty Page Table (DPT) after each action. Assume that DPT holds pageID and recLSN, and XT contains transID and lastLSN

LSN 00:

Transaction Table	
Xact	**lastLSN**
T1	00

Dirty Page Table	
Page	**recLSN**
P7	00

LSN 10:

P9 is brought to buffer and flushed to disk – so the page will not be present in Dirty page table.

Transaction Table	
Xact	**lastLSN**
T1	00
T0	10

Dirty Page Table	
Page	**recLSN**
P7	00

LSN 20:

P8 is brought to buffer and flushed to disk – so the page will not be present in Dirty page table.

Transaction Table	
Xact	**lastLSN**
T1	20
T0	10

Dirty Page Table	
Page	**recLSN**
P7	00

LSN 30:

Indication of checkpoint starting (begin_checkpoint)

LSN 40:

End_checkpoint.

Current contents of transaction table and dirty page table appended to the log. After the end_checkpoint, a master record containing the LSN of the begin_checkpoint log record is written to the stable storage.

Transaction Table	
Xact	**lastLSN**
T1	20
T0	10

Dirty Page Table	
Page	**recLSN**
P7	00

LSN 50:

Transaction Table	
Xact	**lastLSN**
T1	50
T0	10

Dirty Page Table	
Page	**recLSN**
P7	00
P9	50

LSN 60:

Transaction Table	
Xact	**lastLSN**
T1	50
T0	10
T2	60

Dirty Page Table	
Page	**recLSN**
P7	00
P9	50
P6	60

LSN 70:

Transaction Table	
Xact	**lastLSN**
T1	70
T0	10
T2	60

Dirty Page Table	
Page	**recLSN**
P7	00
P9	50
P6	60
P5	70

LSN 80:

P6 is flushed to disk - so the page will not be present in Dirty page table.

Transaction Table	
Xact	**lastLSN**
T1	80

Dirty Page Table	
Page	**recLSN**
P7	00

T0	10
T2	60

P9	50
P5	70

(b) Simulate Analysis phase to reconstruct XT and DPT after the crash. Identify the point where the Analysis phase starts scanning log records and show XT and DPT after each action

In the Analysis phase, the reconstruction of XT and DPT after the crash can be started at checkpoint via end checkpoint record. We scan log forward from this checkpoint.

<u>LSN 30:</u>

The analysis phase starts from the begin checkpoint and initializes the Transaction Table and Dirty Page Table with the value in the end checkpoint log record.

<u>LSN 40:</u> – represents the end check point and before this there are active transactions and hence there is already information in the XT, DPT as below.

Transaction Table	
Xact	**lastLSN**
T1	20
T0	10

Dirty Page Table	
Page	**recLSN**
P7	00

<u>LSN 50:</u>

Transaction Table	
Xact	**lastLSN**
T1	50
T0	10

Dirty Page Table	
Page	**recLSN**
P7	00
P9	50

<u>LSN 60:</u>

Transaction Table	
Xact	**lastLSN**
T1	50
T0	10
T2	60

Dirty Page Table	
Page	**recLSN**
P7	00
P9	50
P6	60

<u>LSN 70:</u>

Transaction Table	
Xact	**lastLSN**
T1	70
T0	10

Dirty Page Table	
Page	**recLSN**
P7	00
P9	50

T2	60

P6	60
P5	70

LSN 80:

Transaction Table	
Xact	**lastLSN**
T1	80
T0	10
T2	60

Dirty Page Table	
Page	**recLSN**
P7	00
P9	50
P5	70

(c) Simulate Redo phase: first identify where the Redo phase starts scanning the log records. Then, for each action identify whether it needs to be redone or not

Redo phase starts scanning forward from the log record containing the smallest recLSN in Dirty Page table.

Dirty Page Table	
Page	**recLSN**
P7	**00**
P9	50
P5	70

LSN	Action
00	Redone
10	Affected page (P9) is in DPT but recLSN (50) > LSN (10) - No Redo
20	Affected page (P8) is not in DPT - No Redo
30, 40	Skipped
50	Redone
60	Affected page (P6) is not in DPT - No Redo
70	Redone
80	Redone