Question 1: Implementing Recovery (1 P.)

Implement the recovery procedures in a mock database environment. In OLAT a template is provided, which gives a mock database, with log.

Given this log and the database, your implementation has to

- Analyze the log to find the loser transactions
- Redo all changes (we assume nothing has been written to disk yet)
- Undo all changes of the loser transactions (this includes writing CLRs)

The log has the same format as shown in the lecture. Use the database class to simulate redoing/undoing operations using the `execute` function. You can simply use the redo/undo operations of the log entries. The template also contains the lecture recovery example log as test case.

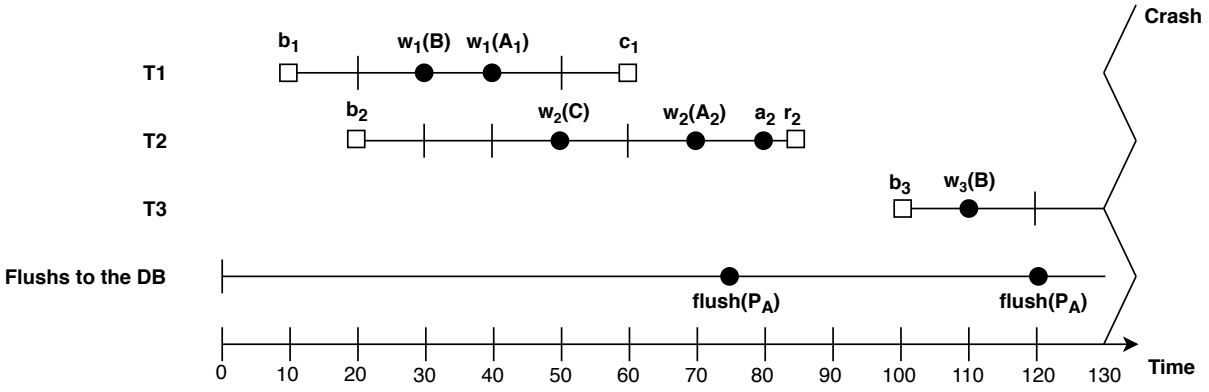
Required submission: Source code; Output after executing the code;

Question 2: Logging and Recovery with Rollback (1 P.)

Given a DBMS with concurrent transactions T_1, T_2, and T_3. These transactions perform the operations illustrated below. The data elements A_1 and A_2 are located on page P_A, B is located on page P_B and C is located in P_C.

T_1 successfully commits at timestamp 60(c_1), while T_2 starts with a abort operation at timestamp 80. The page P_A is flushed at timestamps 75 and 120 from the DB buffer. All rollback operations of T_2 are completed at timestamp 85 (r_2), before b_3 is executed.

During the executions no recovery no checkpoints are set. The recovery is performed by a full REDO.



a) How are the rollback operations performed?

Required submission: Explanation

b) Execute the operations shown in the illustration.

Fill out the table on the next page.

Required submission: Filled out table

- What are the operations of T_2 at timestamp 81/82?
- *Log Entry in Log Buffer*, e.g., [#02, T_1, P_A, $R(A_1)$, $U(A_1)$, 1, 0] (Use $R(\dots)$ / $U(\dots)$ as Redo/Undo information).
- *Log File:* LSNs of the log entries in the log buffer, that are written to the log file.

Time	Operation	DB Buffer		DB Entry		Log Entry in Log Buffer							Log File
		(Page,	LSN)	(Page,	LSN)	[LSN,	TA,	PageID,	Redo,	Undo,	PrevLSN,	UndoNxtLSN]	LSNs
10	b_1					#1	T1	BOT	-	-	-	-	
20	b_2					#2	T2	BOT	-	-	-	-	
30	$w_1(B)$	PB	#3			#3	T1	PB	R(B)	U(B)	#1	-	
40	$w_1(A_1)$	PA	#4			#4	T1	PA	R(A1)	U(A1)	#3	-	
50	$w_2(C)$	PC	#5			#5	T2	PC	R(C)	U(C)	#2	-	
60	c_1					#6	T1	COMMIT	-	-	#4	-	#1, #3, #4, #6
70	$w_2(A_2)$					#7	T2	PA	R(A2)	U(A2)	#5	-	
75	flush(P_A)			PA	#8								
80	a_2					#9	T2	ABORT	-	-	#7	-	
81						<#7'	T2	PA	U(A2)	-	#7	#5>	
82						<#5'	T2	PC	U(C)	-	#7'	#2>	
85	r_2					<#2'	T2	BOT	-	-	#5'	0>	
100	b_3					#13	T3	BOT	-	-	-	-	
110	$w_3(B)$					#14	T3	P(B)	R(B)	U(B)	#13	-	
120	flush(P_A)			PA	#15								
						Crash							

T1

Question 3: Transaction Rollback (1 P.)

a) Suppose that during transaction rollback no log entries are written. Explain what problems will/can arise in this case, by introducing a concrete example. Hint: Consider a data item updated by an aborted transaction, and then updated by a transaction that commits.

Required submission: Explanation

b) Consider transactions that involve interactions with the real world, like the transaction of withdrawing money from an ATM or the transaction sending dismissal notices via postal service. Discuss the feasibility of transaction rollback in such cases. How would the "critical" interactive parts (e.g., releasing the money) of the TA be aligned in time, in order to limit the problematic situations as far as possible?

Required submission: Explanation

Question 4: Schedules - Serializability and Classes (1 P.)

a) Which class does the following schedule have? FSR, VSR, or CSR?

$$s_1 := r_1(c)\ w_2(d)\ w_2(b)\ r_4(c)\ r_5(d)\ r_3(c)\ r_5(c)\ r_4(a)\ r_4(b)\ r_1(d)\ c_5\ r_2(a)\ w_3(c)\ w_1(b)\ r_3(b)\ c_2\ r_3(c)\ w_4(a)\ c_3\ c_1\ c_4$$

Required submission: Class of schedule; Reason for class

b) Given the following schedules, does $s_2 \approx_v s_2'$ hold? Either prove that both schedules are view equivalent or find a counter example.

$$s_2 := w_3(c)\ w_2(a)\ r_3(a)\ r_1(a)\ w_1(b)\ w_3(a)\ w_2(c)\ r_1(a)\ w_3(b)\ c_2\ w_1(c)\ r_3(c)\ c_1\ c_3$$
$$s_2' := r_1(a)\ w_1(b)\ r_1(a)\ w_1(c)\ c_1\ w_2(a)\ w_2(c)\ c_2\ w_3(c)\ r_3(a)\ w_3(a)\ w_3(b)\ r_3(c)\ c_3$$

Required submission: Proof or counterexample

c) Given the following schedule

$$s_4 := w_3(a)\ r_5(a)\ r_2(b)\ r_3(a)\ c_3\ w_2(a)\ r_2(a)\ c_2\ r_5(b)\ r_4(a)\ w_1(b)\ r_1(b)\ c_5\ w_4(c)\ c_4\ w_1(c)\ c_1$$

Create the conflict graph of s_4 and discuss if $s_4 \in CSR$. If yes, reorder s_4 into a serial schedule using the commutativity rules. Does $s_4 \in OCSR$ apply?

Required submission: Conflict graph; Discussion; (If possible) serial schedule and applied rules;