Database Systems WS 2023/24
Exercise 10: Distributed 15.01.2024, Due 22.01.2024 12:00 MEZ
Submitted by Erik Schwede, Suma Keerthi

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.

Submit the code and the output of running your program.

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 X and X_i are located on page P_X.

T_1 aborts at timestamp 60(a_1), while T_2 successfully commits at timestamp 90. The page P_B is flushed at timestamp 55 from the DB buffer. All rollback operations of T_1 are completed at timestamp 65 (r_1), before b_3 is executed. The system crashes at timestamp 110, leaving T_3 incomplete.

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

1. Explain in detail the steps that have to be performed to roll back a transaction.
2. Execute the transactions as shown in the illustration and add all required rollback operations. Fill out the following table.
   - Which assumptions/rules/principles did you apply for logging?
   - What are the operations of T_1 at timestamp 61/62?
   - *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:*Insert the LSNs of the log entries in the log buffer, that are written to the log file.

Database Systems WS 2023/24
Exercise 10: Distributed 15.01.2024, Due 22.01.2024 12:00 MEZ
Submitted by Erik Schwede, Suma Keerthi

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				
20	$w_1(A)$				
30	b_2				
40	$w_1(B_1)$				
50	$w_2(C_1)$				
55	flush(P_B)				
60	a_1				
61					
62					
65	r_1				
70	b_3				
80	$w_2(B_2)$				
90	c_2				
100	$w_3(C_2)$				
				Crash	

Database Systems WS 2023/24
Exercise 10: Distributed 15.01.2024, Due 22.01.2024 12:00 MEZ
Submitted by Erik Schwede, Suma Keerthi

3 Transaction Rollback (1 P.)

1. 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.

2. 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?

4 Schedules - Serializability and Classes (1 P.)

1. Which class does the following schedule have? FSR, VSR, or CSR?

   $s_1 := r_2(b)\ w_2(b)\ c_2\ w_5(a)\ w_5(b)\ r_3(a)\ r_3(d)\ w_1(b)\ r_3(b)\ r_1(c)\ c_1\ r_3(c)\ r_4(c)\ c_3\ c_5\ w_4(c)\ r_4(d)\ w_4(d)\ c_4$

2. 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.

   $$s_2 := w_2(b)\ r_1(b)\ r_2(c)\ r_3(a)\ r_3(b)\ r_3(a)\ w_1(a)\ w_1(c)\ c_1\ r_2(b)\ w_3(c)\ w_2(c)\ c_2\ c_3$$
   $$s_2' := w_2(b)\ r_2(c)\ r_2(b)\ w_2(c)\ c_2\ r_1(b)\ w_1(a)\ w_1(c)\ c_1\ r_3(a)\ r_3(b)\ r_3(a)\ w_3(c)\ c_3$$

3. Given the following schedule

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

   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.