Handin 9 - 10 ECTS

Exercise (1).

What happens for the following three sequences of events if a multi-version timestamp scheduler is used?

What happens if the scheduler does not maintain multiple versions (but rather follows the single-version timestamp approach?

a. $st_1, st_2, r_1(A), r_2(B), w_2(A), w_1(B)$.

b. $st_1, st_2, st_3, r_1(A), r_2(B), w_1(C), r_3(B), r_3(C), w_2(B), w_3(A)$.

c. $st_1, st_2, st_3, r_1(A), r_3(B), w_1(C), r_2(B), r_2(C), w_3(B), w_2(A)$.

where $w_i(X)$ means that transaction i writes item X, and $r_i(X)$ means that transaction i reads item X, st_i means that transaction i starts.

We shall consider them one by one, evaluating whether single-version and/or multiple-version timestamps work. I will be using tables to describe the process and where it fails

a

Table 1.1 Single-version timestamp for schedule *a*

T1	T2	rTS(A)	rTS(B)	wTS(A)	wTS(B)	logic
r1(A)		1				
	r2(B)		2			
	w2(A)			2		TS(T2) > rTS(A)
w1(B)						TS(T1) < rTS(B), FAIL

Single-version fails when T1 tries to overwrite which has been read "later" by T2. The multiversion dependency wont work either, as T1 attempts to write B which is part of T2's readSet.

b

Table 1.2 Single-version timestamp for schedule *a*

T1	T2	T3	rTS(A)	rTS(B)	rTS(C)	wTS(A)	wTS(B)	wTS(C)	logic
r1(A)			1						
	r2(B)			2					
w1(C)								1	
		r3(B)		3					TS(T3) > rTS(B)
		r3(C)			3				TS(T3) > wTS(C)
	w2(B)								TS(T2) < rTS(B)
									FAIL

The single-version fails again for a similar reason. The multi-version will also fail. T2 tries to overwrite a version (in this case version 0, as there were no prior writes) of B, that had already been read by the younger transaction T3.

c

Table 1.3 Single-version timestamp for schedule *a*

T1	T2	T3	rTS(A)	rTS(B)	rTS(C)	wTS(A)	wTS(B)	wTS(C)	logic
r1(A)			1						
		r3(B)		3					
w1(C)								1	
	r2(B)			3					
	r2(C)				2				TS(T2) > wTS(C)
		w3(B)					3		
	w2(A)					2			

There are no conflicts here, so it is allowed to run in both single and mult-version.

Exercise (2)**.**

Consider the following two transactions:
T1: R(A),W(A),R(B),W(B)
T2: R(A),W(A)
Determine whether the following schedules are non-recoverable or recoverable and if relevant cascadeless or strict. Remember to write the explanation.

a. $r_1(A);w_1(A);r_2(A);r_1(B);w_2(A);w_1(B);C2;C1$

b. $r_1(A);r_2(A);w_1(A);r_1(B);w_1(B);w_2(A);C1;C2$

c. $r_1(A);r_2(A);w_1(A);r_1(B);w_1(B);C1;w_2(A);C2$

Lets go through them one-by-one.

a

This schedule is not recoverable as we have,

$$w_1(A) \to r_2(A).$$

And C_2 happens before C_1. The transaction T2 reads data that has been written by T1, and then commits. If T1 decides to abort, there will be an issue, since it has read data that has not been commited.

b

We can start by noting that it is indeed recoverable as we have no relations,

$$w_i(X) \to r_j(X).$$

To determine whether it is strict, we check to see if there any transactions T_j that read/write a field that was previously written by T_i, before T_i has committed. We have the following,

$$w_1(A) \to w_2(A).$$

Which happens before C_1, and it is therefore not strict. To check cascadeless we determine whether T_j reads any fields previously written T_i before the commit. There are no such instances, so it is cascadeless.

c

This schedule is strict, it is identical to the one before, except the commit happens in time,

$$w_1(A) \rightarrow C_1 \rightarrow w_2(A).$$

This is allowed, and there are no other violating relations.

Exercise (3)**.**

Assume shared locks are requested immediately before each read, and exclusive locks immediately before each write. Unlocks occur immediately after the final action of the respective transaction. List all actions which are denied.

a. $r_1(A), r_2(B), w_1(C), r_3(D), r_4(E), w_3(B), w_2(C), w_4(A), w_1(D)$

b. $r_1(A), r_2(B), w_1(C), w_2(D), r_3(C), w_1(B), w_4(D), w_2(A)$

where the notation is as in exercise 1.

a

The action $w_2(C)$ is denied as T_1 has a lock on C, which it has not yet released, as T_1 hasnt finished yet. The others are allowed, for example $w_1(D)$ is fine as T3 unlocked after $w_3(B)$.

b

The action $r_3(C)$ is not allowed, as T_1 still has a lock on C. $w_4(D)$ is denied aswell, as T_2 has a lock on it.