CMPT 454 Assignment 5: Transactions

This assignment is worth 7% of your final grade.

1 - Definitions

a) Define these terms: **[1 mark each]**

1. deadlock
2. rollback
3. blind write
4. unrecoverable schedule
5. phantom problem
6. dirty read
7. unrepeatable read
8. view equivalent schedule
9. serializable schedule
10. conflict equivalent schedule

2 - Transaction Schedules

Write a schedule for parts (**a**) to (**e**) that satisfies the listed requirements. When reading a transaction the first letter indicates the action (**R**ead or **W**rite), the subscript indicates the transaction that the action belongs to (transaction 1, 2 or 3), and the letter in brackets indicates the data object that the transaction affects (A, B or D). **[2 marks each]**

a) For the transactions given below write a non-serial schedule that is conflict serializable.

T1: $R_1(A)$; $R_1(B)$; $W_1(A)$; $W_1(B)$
T2: $W_2(A)$; $R_2(B)$; $W_2(B)$

b) For the transactions given below write a non-serial schedule that is neither conflict serializable nor view serializable.

T1: $R_1(A)$; $R_1(B)$; $W_1(A)$; $W_1(B)$
T2: $W_2(A)$; $R_2(B)$; $W_2(B)$

c) For the transactions given below write a non-serial, but conflict serializable, schedule, where each transaction processes at least one action before any other transaction's final action. In addition, each write of an object by a transaction should immediately follow the read of that object by the same transaction – i.e. there should be no other actions between a read and a write of the same object by the same transaction.

T1: $R_1(A)$; $W_1(A)$; $R_1(B)$; $W_1(B)$
T2: $W_2(B)$; $R_2(D)$; $W_2(D)$;
T3: $R_3(A)$; $W_3(A)$; $W_3(D)$;

d) For the transactions given below write a non-serial schedule that is *not* conflict serializable but *is* view serializable.

T1: $R_1(A)$; $R_1(B)$; $W_1(A)$; $W_1(B)$
T2: $W_2(A)$
T3: $W_3(A)$

e) Re-write the transaction schedule given below by adding commit (C) actions for each transaction such that the schedule would be recoverable should any of the commit actions be replaced by aborts.

$R_1(A)$; $W_2(A)$; $R_3(A)$; $W_3(A)$; $W_1(A)$

f) Re-write the transaction schedule given below by adding commit (C) actions for each transaction such that the schedule would *not* be recoverable if (at least) one of the commit actions is replaced by an abort.

$R_1(A)$; $W_2(A)$; $R_3(A)$; $W_3(A)$; $W_1(A)$

3 - Locking

Consider the schedule shown below, which is not *conflict serializable*.

$R_1(A)$; $R_3(A)$; $R_2(B)$; $W_2(B)$; $R_2(C)$; $R_1(C)$; $W_1(C)$; $W_2(C)$; $R_1(B)$; $W_1(B)$

a) Draw a precedence graph for the schedule. **[1 mark]**

b) Explain why the schedule is not conflict-serializable, by describing how actions cannot be swapped to result in a serial schedule. **[1 mark]**

c) Rewrite the schedule to show the schedule that would result under a *2PL* (not *Strict*) regime using shared and exclusive locks. Show shared and exclusive lock and unlock actions with *S*, *X* and *U*, a subscript that indicates the transaction, followed by the data object to which the lock applies in parentheses. For example, $S_1(A)$ indicates that transaction 1 is applying a shared lock to object *A*. Assume that objects are locked immediately before the first action on the object and unlocked as soon as possible. **[2 marks]**

d) Give an interleaved (i.e. non-serial) schedule of the transactions given below under a *Strict 2PL* regime that would *not* result in deadlock. Include the lock and unlock actions in your schedule. **[2 marks]**

T1: $R_1(A)$; $W_1(A)$; $R_1(B)$; $W_1(B)$
T2: $R_2(C)$; $W_2(C)$; $R_2(A)$
T3: $R_3(B)$; $R_3(C)$

Your schedule *must* begin with these actions:

$X_1(A)$; $R_1(A)$; $W_1(A)$; $X_2(C)$; $R_2(C)$; $W_2(C)$

e) Give an interleaved schedule of the transactions shown in part (**d**) that *would* result in deadlock under a Strict 2PL regime. Include the lock actions in your schedule, ending with the lock actions that result in deadlock. **[2 marks]**

Your schedule *must* begin with these actions:

$X_1(A)$; $R_1(A)$; $W_1(A)$; $X_2(C)$; $R_2(C)$; $W_2(C)$

f, g and h) Consider the schedule shown below, which results in *deadlock*.

$R_1(A)$; $W_1(A)$; $W_2(B)$; $R_3(C)$; $R_4(D)$; $W_4(D)$; $W_2(D)$; $R_3(B)$; $R_4(A)$; $W_4(A)$; $R_1(C)$; $W_1(C)$

f) Draw the waits-for graph for the schedule **[2 marks]**

g) Which transaction(s) would be aborted in a *wound-wait* scheme? Assume that priority is given to lower numbered transactions and that transactions commit as soon as they have performed their last action. **[1 mark]**

h) Which transaction(s) would be aborted in a *wait-die* scheme? Assume that priority is given to lower numbered transactions and that transactions commit as soon as they have performed their last action. **[1 mark]**

4 – Optimistic Concurrency Control

Describe what happens for each of the schedules (**a**) to (**d**) shown below. Note whether each transaction completes, is aborted or is suspended. Where a transaction is waiting (i.e. is suspended) explain what the transaction is waiting for and what will happen to the transaction based on results of the action it is waiting for. Assume that the commit bit for all data objects is true before the schedule commences. When reading a schedule, the first letter indicates the action (**S**tart-time, **R**ead, **W**rite or **C**ommit), the subscript indicates the transaction that the action belongs to (transaction 1, 2 or 3), and the letter in brackets indicates the data object that the transaction affects (A, B or D) **[3 marks each]**

a) S_1; S_2; $R_1(A)$; $W_1(A)$; $R_2(B)$; $W_2(A)$; $W_2(B)$; $W_1(B)$

b) S_1; S_2; $R_1(A)$; $W_1(A)$; $W_2(B)$; $R_1(B)$; $W_1(B)$

c) S_1; S_2; $R_1(A)$; $W_1(A)$; $R_2(A)$; $W_2(A)$; $R_1(B)$; $W_1(B)$; $R_2(B)$; $W_2(B)$

d) S_1; S_2; $R_1(A)$; $W_2(A)$; C_2; $W_1(A)$

Assessment

The assignment is out of 46. Marks are assigned as follows:

- Question 1 – 10
- Question 2 – 12
- Question 3 – 12
- Question 4 – 12

Submission

The assignment is due by 11:59pm on Friday the 7th of August.

CMPT 454 Home

John Edgar (johnwill@sfu.ca)