CSC553: Homework 5

Due: Jun 6^{th}, 2022

This assignment is on concurrency control and recovery.

1 Pessimistic Concurrency Control

1.1 Two-Phase Locking

Consider the following schedule consisting of three transactions, T_1, T_2, and T_3:

Insert Shared (S) and Exclusive (X) Lock and Unlock (U) actions to determine whether the schedule can be realized with 2PL protocol.

T_1	T_2	T_3
	R(A)	
	W(B)	
	R(B)	
W(B)		
	W(A)	
		R(B)
R(C)		
		W(C)

1.2 Deadlocks

Consider the following two transactions T_1 and T_2.

T_1

```
read(A);
read(B);
if (A = 0)
begin
B := B + 1;
write(B);
end;
```

T_2

```
read(B);
read(A);
if (B = 0)
```

```
begin
A := A + 1;
write(A);
end;
```

(i) Add Lock and Unlock statements to these two transactions.
(ii) Draw a wait-for graph, and state if there is a possibility of a deadlock.

1.3 Strict 2PL

Consider the following transaction schedule where C stands for commit.

T_1	T_2	T_3
	R(B)	
R(A)		
		R(A)
W(B)		
	R(A)	
		R(B)
		W(B)
		C
	C	
C		

(i) Is this schedule possible with 2PL? If so, show how this schedule is executed with two-phase locking.

(ii) Is this schedule feasible under **strict** two-phase locking? Why or why not?

(iii) Is this schedule recoverable? Why or why not?

2 Isolation Levels

Consider the following table XboxGames(name, price) and assume that these values already exist in the database ('okGame', 40), ('goodGame', 50), ('AWESOMEGame', 60). We have the following two transactions:

T1
BEGIN TRANSACTION
UPDATE XboxGames SET price=22 WHERE name='okgame';
INSERT INTO XboxGames VALUES ('BADGame', 0);
UPDATE XboxGames SET price=38 WHERE name='okGame';
COMMIT;

T2
BEGIN TRANSACTION SET TRANSACTION ISOLATION LEVEL UNCOMMITTED
SELECT AVG(price) AS averagePrice FROM XboxGames
COMMIT;

Above two transactions are hitting the DBMS roughly at the same time. What are the possible values for averagePrice?

3 Optimistic Concurrency Control

3.1 Timestamp-based CC

Consider the following schedule. Explain what happens when transactions try to execute as per this schedule and the DBMS uses timestamp-based concurrency control. We use ST to denote the start of a transaction, C for commit, A for abort. Please use D to denote any delays.

$ST_1 \rightarrow ST_2 \rightarrow ST_3 \rightarrow ST_4 \rightarrow R_2(X) \rightarrow R_1(X) \rightarrow W_2(X) \rightarrow W_4(X) \rightarrow W_1(X) \rightarrow C_1 \rightarrow W_3(X) \rightarrow A_4 \rightarrow R_2(Y) \rightarrow W_2(Y) \rightarrow R_3(Y) \rightarrow C_2 \rightarrow W_3(Y) \rightarrow C_3$

Answer (Fill in the table below showing what happens as the transactions execute):

T_1	T_2	T_3	T_4	X	Y
1	2	3	4	RT = 0 WT = 0 C = true	RT = 0 WT = 0 C = true
	$R_2(X)$				

3.2 Multiversion Concurrency Control

Consider the following schedule. Explain what happens when transactions try to execute as per this schedule and the DBMS uses multiversion concurrency control:

$ST_1 \rightarrow ST_2 \rightarrow ST_3 \rightarrow ST_4 \rightarrow R_1(X) \rightarrow R_3(X) \rightarrow W_3(X) \rightarrow R_2(X) \rightarrow R_4(X) \rightarrow W_2(X) \rightarrow W_4(X)$

(Fill in the table below showing what happens as the transactions execute):

T_1	T_2	T_3	T_4	X_0	...
1	2	3	4	RT = 0 WT = 0 C = true	RT = 0 WT = 0 C = true
$R_1(X)$				RT=1	

4 Recovery

The following is a sequence of undo-log records written by two transactions T and U:
$< START\ T >$
$< T, A, 10 >$
$< START\ U >$
$< U, B, 20 >$
$< T, C, 30 >$
$< U, D, 40 >$
$< COMMIT\ U >$
$< T, E, 50 >$
$< COMMIT\ T >$

Describe the action of the recovery manager by stating which database elments will be updated with which value if there is a crash and the last log record to appear on disk is:

(i) $< START\ U >$
(ii) $< C0MMIT\ U >$
(iii) $< T, E, 50 >$
(iv) $< C0MMIT\ T >$