Homework Assignment 3

Databases II

**Transactions and Schedules**

Exercise 1. Answer the following questions considering schedule S. For the yes/no questions, briefly explain and justify your answer.

S = T1 :W(A), T3 :R(B), T2 :W(A), T2 :W(B), T1 :W(B), T1 :Commit, T3 :W(C), T2 :R(A), T2 :Commit, T3 :Commit

1. Decide whether S is strict, avoid cascade abort (cascadeless), and recoverable? (You must give one yes/no answer for each class.) (20 marks)

Strict: No, because T2 starts before T1 commits.

Cascadeless: No, because after T3 read T1 and T2 have yet to commit.

Recoverable: Yes, because there are commits after read changes.

1. Decide whether S is conflict-serializable and view serializable. (You must give one yes/no answer for each class.) (10 marks)

Conflict-serializable: No, because T1 and T2 are different transactions using the same data so there is a conflict.

View serializable: No, because T1 and T2 have their writes first while T3 has a read first.

1. Draw the precedence graph for S. (5 marks)

T3

T2

T1

4. If S is not conflict serializable, explain changes in the schedule (without changing the actions in the transactions) to make it conflict serializable. (10 marks)

By allowing T1 W(B) before T2 W(B), the schedule will be conflict serializable.

**Locking Processes**

Exercise 2. Define strict schedules. Explain why applying strict 2PL results in a strict schedule, but applying 2PL does not necessarily give a strict schedule. (10 marks)

A strict schedule is when a transaction has to fully commit or abort before another transaction can begin. Applying strict 2PL results in a strict schedule due to the fundamental rules that strict 2PL follows. In strict 2PL a transaction does not release its lock until it commits whereas in 2PL a transaction has the ability to release locks before a commit. This results in the option of 2PL not necessarily giving a strict schedule.

Exercise 3. Explain the phantom problem when a database shrinks or grows. Explain why strict 2PL with locking records can cause the phantom problem, but locking tables in strict 2PL solves the phantom problem. (10 marks)

A phantom problem is when consecutive execution of the same query give varying results in the number of rows. Strict 2PL with locking records can cause the phantom problem because although it ensures conflict serializability, when data is added or deleted that serializability is broken and results from the same query will output different results. Locking tables in strict 2PL would solve this issue because then data in the table cannot be altered and therefore queries would return the same results every time.

Exercise 4. Give an example of the phantom problem when a database shrinks (some records are deleted). (10 marks, bonus)

A phantom problem can be demonstrated by using bank account with 2 users. If user 1 requests to withdrawal $1500 from their bank account with a balance of $1000, the user would not be able to complete the transaction because of insufficient funds. Meanwhile user 2 makes a deposit of $1000. Now if user 1 tries to make the same withdrawal of $1500, the transaction will be successful. This is an example of a phantom read as consecutive execution of the same query returns a different result due to the criteria not being initially met.

Exercise 5. Answer the following questions considering the sequence of actions, T1 :R(A), T2 :R(B), T2 :W(B), T3 :R(B), T1 :W(C), T1 :Commit, T2 :R(D), T2 :Commit, T3 :Commit.

1. Apply strict 2PL and 2PL by extending the sequence with shares and exclusive locks (e.g., add T1 : S(A) and T1 : X(A) respectively when T1 must request a shared lock and an exclusive lock on A). (10 marks)

STRICT 2PL:

T1:S(A), T1:R(A), T1:U(A), T2:S(B), T2:X(B), T2:R(B), T2:W(B), T2:U(B), T2:S(D), T2:R(D), T2:U(D), T2:Commit, T3:S(B), T3:R(B), T3:U(B), T1:S(C), T1:X(C), T1:W(C), T1:U(C), T1:Commit, T3:Commit.

2PL:

T1:S(A), T1:R(A), T1:U(A), T2:S(B), T2:X(B), T2:R(B), T2:W(B), T2:U(B), T3:S(B), T3:R(B), T3:U(B), T1:S(C), T1:X(C), T1:W(C), T1:U(C), T1:Commit, T2:S(D), T2:R(D), T2:U(D), T2:Commit, T3:Commit.

1. Explain why the schedule obtained from applying strict 2PL in (1) is strict. Decide whether the schedule obtained from applying 2PL in (1) is strict, and justify your answer. (10 marks)

The schedule obtained from applying strict 2PL is strict because all the transaction unlock their locks and commit before another transaction accesses that same data. The schedule obtained from applying 2PL is not strict because T2 and T3 preform reads and write on data that has yet to be committed by T1.

3. Draw the wait-for graphs for the schedules in (2) and decide whether the schedules cause deadlocks? (10 marks)

STRICT 2PL : 2PL :

T3

T2

T1

T3

T2

T1

**Crash Recovery**

Exercise 6. Explain why DBMSs implement a no-force steal approach. Describe a challenge in crash recovery caused by this approach that does not happen with the force no-steal approach. (10 marks)

A no-force steal approach is when it is allowed for an update to be written to a disk before a commit. DBMSs implement this to optimize performance as the cost of large transaction can be costly. This approach comes with the challenge of an amplified number of rollback procedures.

Exercise 7. Answer the following questions considering the log records in Figure 1. In each case, justify your answer.

1. Assuming that locking is applied for pages, does strict 2PL allow the schedule in these log records? (10 marks)

Strict 2PL would not allow the schedule in Figure 1 because the locks are not released when the transaction is complete.

1. Which transaction is a loser transaction? (5 marks)

T1 and T3 are loser transaction as they do not commit.

1. Can ARIES recover the database from this crash while preserving the ACID properties? If the answer is no, which ACID property cannot be guaranteed? (10 marks)

ARIES cannot recover the database from the crash while preserving the ACID properties. The two ACID properties that cannot be guaranteed are consistency and isolation. The data does not maintain a constant state from start to finish of each transaction and each transaction is not executed independently.