**2: Transaction Management**

**Q2(a)**: Consider a relation R(A) containing {(5),(6)} and two transactions:

**T1**: Update R set A = A+1;

**T2**: Update R set A = 2\*A.

Suppose both transactions are submitted under the **isolation and atomicity** properties. Provide all possible final states of R (values of R) by considering all possible serializable and non-serializable schedules.

1. Considering all possible serializable schedules.

Serializable T1; T2: final states of R is {(12),(14)}

|  |  |  |  |
| --- | --- | --- | --- |
| Time | T1 | T2 | A |
| t0 | write(A) |  | {(5),(6)} |
| t1 | commit |  | {(6),(7)} |
| t2 |  | write(A) | {(6),(7)} |
| t3 |  | commit | {(12),(14)} |

Final states of R is {(12),(14)}

Serializable T2: T1:

|  |  |  |  |
| --- | --- | --- | --- |
| Time | T2 | T1 | A |
| t0 | write(A) |  | {(5),(6)} |
| t1 | commit |  | {(10),(12)} |
| t2 |  | write(A) | {(10),(12)} |
| t3 |  | commit | {(11),(13)} |

Final states of R is {(11),(13)}

1. Considering all possible non-serializable schedules.

Non-Serializable T1; T2: final states of R is {(6),(7)}

|  |  |  |  |
| --- | --- | --- | --- |
| Time | T1 | T2 | A |
| t0 | write(A) |  | {(5),(6)} |
| t1 |  | write(A) | {(5),(6)} |
| t2 |  | commit | {(10),(12)} |
| t3 | commit |  | {(6),(7)} |

Non-Serializable T2; T1: final states of R is {(10),(12)}

|  |  |  |  |
| --- | --- | --- | --- |
| Time | T1 | T2 | A |
| t0 |  | write(A) | {(5),(6)} |
| t1 | write(A) |  | {(5),(6)} |
| t2 | commit |  | {(6),(7)} |
| t3 |  | commit | {(10),(12)} |

**Q2(b)** Consider a table R(A) containing {(1),(2)} and following two transactions run concurrently:

**T1**: Update R set A = 2\*A; commit;

**T2**: Select avg(A) from R; commit;

If transaction T2 executes using **READ UNCOMMITTED**, what are the possible average values of T2 returns? Provide an explanation for each value it returns.

If transaction T2 executes using read uncommitted, if T1 commited before T2 read, the result of avg(A) is 3, the step is like following table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| time | T1 | T2 | A’s value | avg(A) |
| t0 | write(A) |  | {(1),(2)} |  |
| t1 | commit; |  | {(2),(4)} |  |
| t2 |  | read(A) | {(2),(4)} | 3 |
| t3 |  | commit; | {(2),(4)} | 3 |

If transaction T2 executes using read uncommitted, if T1 begin before T2 and commited after T2 read, the result of avg(A) is 3, When T2 read A and compute avg(A) then get (2+4)/2=3. Because T2 read the data of T1 written but has not committed.

the step is like following table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| time | T1 | T2 | A’s value | avg(A) |
| t0 | write(A) |  | {(2),(4)} |  |
| t1 |  | read(A) | {(2),(4)} | 3 |
| t2 |  | commit; | {(2),(4)} | 3 |
| t3 | commit; |  | {(2),(4)} |  |

If transaction T2 executes using read uncommitted, if T1 begin after T2 and commited before T2 read, the result of avg(A) is 1.5, the step is like following table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| time | T1 | T2 | A’s value | avg(A) |
| t0 |  | read(A) | {(1),(2)} | 1.5 |
| t1 |  | commit; | {(1),(2)} | 1.5 |
| t2 | write(A) |  | {(2),(4)} |  |
| t3 | commit; |  | {(2),(4)} |  |

If transaction T2 executes using read uncommitted, if T1 begin after T2 and commited after T2 read, the result of avg(A) is 1.5, the step is like following table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| time | T1 | T2 | A’s value | avg(A) |
| t0 |  | read(A) | {(1),(2)} | 1.5 |
| t1 | write(A) |  | {(2),(4)} |  |
| t2 | commit; |  | {(2),(4)} |  |
| t3 |  | commit; | {(2),(4)} | 1.5 |

**Q2(c)** Consider table R(A) containing {(1),(2)}. Suppose we have the following two transactions running concurrently:

**T1:** Update T set A=2\*A;

insert into R values (6);

Commit;

**T2:** Select avg(A) from R;

Select avg(A) from R ;

Commit;

If transaction T2 executes using **REPEATABLE READ,** what are the possible values returned by T2 in its SECOND select statement? Provide an explanation for each value it returns.

Condition1: Using repeatable read, if T2 is scheduled after T1, the result of avg(A) is 4. The reason is as following table shows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| time | T1 | T2 | A | avg(A) |
| t0 | write(A) |  | {(1),(2)} |  |
| t1 | commit |  | {(2),(4),(6)} |  |
| t2 |  | read(A) | {(2),(4),(6)} | 4 |
| t3 |  | read(A) | {(2),(4),(6)} | 4 |
| t4 |  | commit |  |  |

Condition2: Using repeatable read, if T2 is scheduled but T1 has not committed, the result of avg(A) is 1.5. The reason is as following table shows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| time | T1 | T2 | A | avg(A) |
| t0 | write(A) |  | {(1),(2)} |  |
| t1 |  | read(A) | {(1),(2)} | 1.5 |
| t2 |  | read(A) | {(1),(2)} | 1.5 |
| t3 |  | commit |  |  |
| t4 | commit |  | {(2),(4),(6)} |  |

Condition3: Using repeatable read, if T2 is scheduled and T1 writes something and committed, T2’ first read is before T1’s committed and T2’s second read is after T1’s committed , the result of avg(A) is 1.5. The reason is as following table shows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| time | T1 | T2 | A | avg(A) |
| t0 | write(A) |  | {(1),(2)} |  |
| t1 |  | read(A) | {(1),(2)} | 1.5 |
| t2 | commit |  | {(2),(4),(6)} |  |
| t3 |  | read(A) | {(2),(4),(6)} | 1.5 |
| t4 |  | commit | {(2),(4),(6)} |  |

Condition4: Using repeatable read, if T2 is scheduled before T1,, the result of avg(A) is 1.5. The reason is as following table shows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| time | T1 | T2 | A | avg(A) |
| t0 |  | read(A) | {(1),(2)} | 1.5 |
| t1 |  | read(A) | {(1),(2)} | 1.5 |
| t2 |  | commit | {(1),(2)} |  |
| t3 | write(A) |  | {(2),(4),(6)} |  |
| t4 | commit |  | {(2),(4),(6)} |  |

**Q2(d)** For each of the following schedules of read, write, commit and abort actions done by transactions T1, T2, T3, state whether they are recoverable or not. If not recoverable, what type of inconsistencies might we have in the final database state? If recoverable, which of the other transactions need to be rolled back? Note ri (A) and wi (A) mean that transaction Ti reads and writes database object A, respectively.

|  |  |  |
| --- | --- | --- |
| **S1** | | |
| T1 | T2 | T3 |
| w1(A);  w1(B);  abort; | r2(A);  w2(A\*2)  r2(B);  commit; | r3(B);  r3(B+5);  commit; |

S1 is not recoverable because T2 read the data A which T1 written A but not commited. which violates if a transaction T2 commits and has read item A that was written before by a different transaction T1 and T2 has not committed.

The final database may 出现丢失更新问题，because T1 abort after T2 commited, which T1’s update on A may be lost.

|  |  |  |
| --- | --- | --- |
| **S2** | | |
| T1 | T2 | T3 |
| r1(A);  r1(B);  w1(B);  abort; | r2(A);  w2(A\*2) | r3(B);  w3(B+5); |

S2 is recoverable, 虽然T1’s written on B has not commited but T3 read on B is also not commited.

if recoverable, T1, T2, T3 all need to roll back.