CSC 370

Activity Worksheet: Transaction Isolation Levels

Mr. Yichun Zhao

Fall 2022

This worksheet should provide practice questions for better understanding the impact of isolation levels on concurrency in databases. In these questions, you are given a set of transactions that will execute concurrently. You should specify all of the possible database end states for each of the three isolation levels: read uncommitted, read committed, and read repeatable. Assume the following initial state:

$R(x, y) = \{(1, 1), (2, 1), (3, 3)\}$
$S(y, z) = \{(3, 1), (1, 2), (0, 2)\}$.

Questions

1.
```
START TRANSACTION; -- t1
    SELECT @y := y FROM R WHERE x = 3; -- query1
    UPDATE S SET z = 1 WHERE y = @y; -- query2
COMMIT;

START TRANSACTION; -- t2
    SELECT @y := y FROM R WHERE x = 3; -- query3
    UPDATE S SET z = 2 WHERE y = @y; -- query4
COMMIT;
```

Solution:

2.

```
START TRANSACTION; -- t1
    SELECT @y := y FROM R WHERE x = 3; -- query1
    UPDATE S SET z = z + @y WHERE y = @y; -- query2
COMMIT;

START TRANSACTION; -- t2
    SELECT @y := y FROM R WHERE x = 3; -- query3
    UPDATE S SET z = z + @y WHERE y = @y; -- query4
COMMIT;
```

Solution:

3.
```
START TRANSACTION; -- t1
    SELECT @y := y FROM R WHERE x = 3; -- query1
    UPDATE S SET z = z + @y WHERE y = @y; -- query2
    UPDATE R SET x = NULL WHERE x = 3; -- query3
COMMIT;

START TRANSACTION; -- t2
    SELECT @y := y FROM R WHERE x = 3; -- query4
    UPDATE S SET z = z + @y WHERE y = @y; -- query5
    UPDATE R SET x = 4 WHERE x = 3; --query6
COMMIT;
```

Solution:

Solutions

Question 1

Both transactions only modify S and only by setting it to a constant, one of which is its initial value; so, the database "end state" after the two transactions execute can only be one of two options (difference emphasised):

$S(y, z) = \{(3, \mathbf{1}), (1, 2), (0, 2)\}$
$S(y, z) = \{(3, \mathbf{2}), (1, 2), (0, 2)\}$

Let's work through each possible ordering of the queries and identify the result DB and the isolation levels in which it could occur:

< query1, query2, query3, query4 >
This is a serialised execution of the transactions (t1 occurred entirely before t2), so could occur at any isolation level.

The result will be result 2, as the last statement to execute will be the one that sets S.z=2.

< query3, query4, query1, query2 >
This is also a serialised execution of the transactions (t2 occurred entirely before t1), so could occur at any isolation level.

The result will be result 1, as the last statement to execute will be the one that sets S.z=1.

< query1, query3, query4, query2 >
Here, t2 occurred in the middle of t1. However, the write to S.z by t2 does not affect the read of R.y by t1. So, both transactions could take-out read-only locks on R.y. This could occur at an isolation level of repeatable read or lower.

The result will be result 1, as the last statement to execute will be the one that sets S.z=1, overwriting query 4.

< query1, query3, query2, query4 >
Here, queries for t1 and t2 were alternated. However, as above, the writes to S.z are not affected by the reads of R.y. So, both transactions could take-out read-only locks on R.y. This could occur at an isolation level of repeatable read or lower.

The result will be result 2, as the last statement to execute will be the one that sets S.z=2, overwriting query 2.

Other permutations are the same as above, just with the role of t1 and t2 reversed.

Question 2

These transactions are more complex, so let's convert them explicitly to a series of reads and writes

Transaction1
Read (R.x, R.y) –r1
Read (S.y, S.z) –r2
Write S.z –w1

Transaction2
Read (R.x, R.y) –r3
Read (S.y, S.z) –r4
Write S.z –w2

Observe too that the write uses the previously read values from R.y and S.z. That is the primary difference compared to the previous question. Let's work through different read/write sequences. For simplicity, we will assume that w1 occurs before w2 (the logic is similar if reversed).

< r1, r2, w1, r3, r4, w2 >
This is a fully serialised execution sequence that could happen at any isolation level. The result will be $S(y, z) = \{(3, 7), (1, 2), (0, 2)\}$.

< r1, r2, r3, w1, r4, w2 >
This is not serialised, but w1 does not affect the result of r3, as these are on different tables. We could achieve this sequence with a read-only shared lock. It could occur at a repeatable read or lower isolation level. The result will again be $S(y, z) = \{(3, 7), (1, 2), (0, 2)\}$.

< r1, r2, r3, r4, w1, w2 >
In this case, r4 is a dirty read; the value is changed by w1 after r4 and before it is used by w2. It can only occur at an isolation level of read uncommitted. The result will be $S(y, z) = \{(3, 4), (1, 2), (0, 2)\}$, as w2 overwrites the change made by w1.

Question 3

Again, we have a quite complex one, so let's explicitly indicate the reads and writes

Transaction1
Read (R.x, R.y) –r1
Read (S.y, S.z) –r2
Write (S.z) –w1
Read (R.x) –r3
Write (R.x) –w2

Transaction2
Read (R.x, R.y) –r4
Read (S.y, S.z) –r5
Write (S.z) –w3
Read (R.x) –r6
Write (R.x) –w4

Compared to the previous question, now we also write to R.x, the value read at the start of the transaction.

< r1, r2, w1, r3, w2, r4, r5, w3, r6, w4 >
This corresponds to a fully serialised execution, which could happen at any isolation level. The result is $R(x, y) = \{(1, 1), (2, 2), (NULL, 3)\}$ and $S(y, z) = \{(3, 4), (1, 2), (0, 2)\}$.

< r1, r2, w1, r3, r4, w2, r5, w3, r6, w4 >
Observe that this already creates a dirty read, as w2 changes the value read by r4. Thus, this execution sequence is only possible with an isolation level of read uncommitted. In fact, there are no non-serialised execution orders for these transactions that does not involve a dirty read. This leads to double-incremented S.z, because both transactions set @y successfully:$R(x, y) = \{(1, 1), (2, 2), (4, 3)\}$ and $S(y, z) = \{(3, 7), (1, 2), (0, 2)\}$.

< r1, r2, w1, r3, r4, r5, w3, r6, w4, w2 >
In this case, we have multiple dirty reads, which coincidentally cancel each other out. Although t1 and t2 are going to double-increment S.z, one of them overwrites the other's changes and we end up with the same result as serialised execution: $R(x, y) = \{(1, 1), (2, 2), (NULL, 3)\}$ and $S(y, z) = \{(3, 4), (1, 2), (0, 2)\}$.

Observe that there are two other possibilities that can be obtained by swapping t1 and t2 in the above two cases:

$R(x, y) = \{(1, 1), (2, 2), (NULL, 3)\}$ and $S(y, z) = \{(3, 7), (1, 2), (0, 2)\}$
$R(x, y) = \{(1, 1), (2, 2), (4, 3)\}$ and $S(y, z) = \{(3, 4), (1, 2), (0, 2)\}$