Transactions Exercises

1. Consider two tables R(A,B) and S(C). Below are pairs of transactions. For each pair, decide whether it is possible for nonserializable behavior to be exhibited when executing the transactions concurrently, while respecting their specified isolation levels. Assume individual statements are executed atomically, and each transaction executes to completion.

(a) Transaction 1:

```
Set Transaction Isolation Level Read Committed;
Select count(*) From R;
Select count(*) From S;
Commit;
```

Transaction 2:

```
Set Transaction Isolation Level Serializable;
Insert Into R Values (1,2);
Insert Into S Values (3);
Commit;
```

(b) Transaction 1:

```
    Set Transaction Isolation Level Read Committed;
    Select count(*) From R;
    Select count(*) From S;
    Commit;
```

Transaction 2:

```
    Set Transaction Isolation Level Serializable;
    Insert Into R Values (1,2);
    Insert Into R Values (3,4);
    Commit;
```

(c) Transaction 1:

```
    Set Transaction Isolation Level Repeatable Read;
    Select count(*) From R;
    Select count(*) From S;
    Select count(*) From R;
    Commit;
```

Transaction 2:

```
    Set Transaction Isolation Level Serializable;
    Insert Into R Values (1,2);
    Commit;
```

2. Consider table Item(name,price) where name is a key, and the following two concurrent transactions.

T1:

```
  Begin Transaction;
  S1: Insert Into Item Values ('scissors',40);
  S2: Update Item Set price = price + 30 Where name =
'pencil';
  Commit;
```

T2:

```
  Begin Transaction;
  S3: Select Avg(price) As a1 From Item;
  S4: Select Avg(price) As a2 From Item;
  Commit;
```

Assume that the individual statements S1, S2, S3, and S4 always execute atomically. Suppose initially there are two tuples in Item: (pencil,20) and (pen,30). Each transaction runs once and commits. Transaction T1 always executes with isolation level *Serializable*.

(a) If transaction T2 executes with isolation level *Serializable*, what possible pairs of values a1 and a2 are returned by T2?

(b) If transaction T2 executes with isolation level *Repeatable-Read*, what possible pairs of values a1 and a2 are returned by T2?

(c) If transaction T2 executes with isolation level *Read-Committed*, what possible pairs of values a1 and a2 are returned by T2?

(d) If transaction T2 executes with isolation level *Read-Uncommitted*, what possible pairs of values a1 and a2 are returned by T2?

3. Consider table Person(name,age) and the following transaction T:

```
Begin Transaction;
Q1: Select Avg(age) From Person;
<read-only activity>
Q2: Select Avg(age) From Person;
Commit;
```

Assume queries Q1 and Q2 always execute atomically.

(a) Suppose all other transactions in the system are declared as *Serializable* and Read-Only. What is the weakest isolation level needed for transaction T to ensure that queries Q1 and Q2 will always get the same result? Choices are: *Read-Uncommitted*, *Read-Committed*, *Repeatable-Read*, *Serializable*

(b) Suppose all other transactions in the system are declared as *Serializable*, and they only involve queries, updates, and deletions. What is the weakest isolation level needed for transaction T to ensure that queries Q1 and Q2 will always get the same result? Choices are: *Read-Uncommitted*, *Read-Committed*, *Repeatable-Read*, *Serializable*

(c) Suppose all other transactions in the system are declared as *Serializable*, and we know nothing else about them. What is the weakest isolation level needed for transaction T to ensure that queries Q1 and Q2 will always get the same result? Choices are: *Read-Uncommitted*, *Read-Committed*, *Repeatable-Read*, *Serializable*

Now consider the following variation, where the two queries are in two different transactions:

T1:

```
Begin Transaction;
Q1: Select Avg(age) From Person;
<read-only activity>
Commit;
```

T2:

```
Begin Transaction;
Q2: Select Avg(age) From Person;
<read-only activity>
Commit;
```

(d) Suppose both transactions T1 and T2 execute with isolation level *Serializable*. Consider scenarios (a), (b), and (c) above for all other transactions in the system. For which of these scenarios, if any, are we guaranteed to always get the same result for Q1 and Q2?

4. Consider table Worker(name,pay) where name is a key, and the following two concurrent transactions.

T1:

```
Begin Transaction
S1: update Worker set pay = 2*pay where name = 'Amy'
S2: update Worker set pay = 3*pay where name = 'Amy'
Commit
```

T2:

```
Begin Transaction
S3: update Worker set pay = pay-20 where name = 'Amy'
S4: update Worker set pay = pay-10 where name = 'Amy'
Commit
```

Assume that the individual statements S1, S2, S3, and S4 always execute atomically. Let Amy's pay be 50 before either transaction begins execution.

(a) Suppose both transactions T1 and T2 execute to completion with isolation level *Serializable*. What are the possible values for Amy's final pay?

(b) Suppose both transactions T1 and T2 execute to completion with isolation level *Read-Committed*. What are the possible values for Amy's final pay?

(c) Suppose transaction T1 executes with isolation level *Read-Committed*, transaction T2 executes with isolation level *Read-Uncommitted*, and both transactions execute to completion. What are the possible values for Amy's final pay?

(d) Suppose both transactions T1 and T2 execute to completion with isolation level *Read-Uncommitted*. What are the possible values for Amy's final pay?

(e) Suppose both transactions T1 and T2 execute with isolation level *Serializable*. Transaction T1 executes to completion, but transaction T2 rolls back after statement S3 and does not re-execute. What are the possible values for Amy's final pay?

Hide Answers

1.
(a) Yes, nonserializable behavior is possible (two statements of Transaction 1 execute before and after Transaction 2, respectively)
(b) No, nonserializable behavior is not possible (state of S is same before and after Transaction 2)
(c) Yes, nonserializable behavior is possible (first and third statements of Transaction 1 execute before and after Transaction 2, respectively)

2.
(a) (25,25) (40,40)
(b) (25,25) (40,40)
(c) (25,25) (25,40) (40,40)
(d) (25,25) (25,30) (25,40) (30,30) (30,40) (40,40)

3.

(a) *Read-Uncommitted*
(b) *Repeatable-Read*
(c) *Serializable*
(d) Only (a)

4.

(a) 120 270
(b) 120 270
(c) 120 210 270
(d) 120 150 170 210 230 270
(e) 300