LAB - 9,10

2. Simulate an environment that allows multithreaded database access for multi-transaction handling

```c
#include <stdio.h>
#include <pthread.h>
#include <time.h>
#include <stdlib.h>

int database = 0;
void* transaction(int id){
    printf("transaction %d has started\n", id);
    sleep(5);
    database++;
    printf("Database is updated to the value %d now\n", database);
    sleep(5);
    printf("transaction %d has ended\n", id);
}

int main(){
    pthread_t t1, t2,t3,t4;
    pthread_create(&t1, NULL, transaction, (void*)1);
    pthread_create(&t2, NULL, transaction, (void*)2);
    pthread_create(&t3, NULL, transaction, (void*)3);
    pthread_create(&t4, NULL, transaction, (void*)4);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    pthread_join(t3, NULL);
    pthread_join(t4, NULL);
    return 0;
```

Output:

```
sandy@LAPTOP-BG8QH1CU:/mnt/c/Users/sandi/OneDrive/Documents/Coding/DSA$ ./w
transaction 1 has started
transaction 3 has started
transaction 2 has started
transaction 4 has started
Database is updated to the value 1 now
Database is updated to the value 4 now
Database is updated to the value 2 now
Database is updated to the value 3 now
transaction 3 has ended
transaction 4 has ended
transaction 2 has ended
transaction 1 has ended
```

Demo

3) Simulate a deadlock situation derived from multiple transactions attempting to access & update the same tuple

SQLQuery2.sql - (l...RAGATI SINHA (64))* | SQLQuery1.sql - (l...RAGATI SINHA (62))*

```
begin transaction
update Book_Details set Book = 'Emotion machine part 2' where Book_ID = 'Mi009_Emo';
update Book_Details set Book = 'Self comes to your mind' where Book_ID = 'Da001_Sel';
rollback transaction
commit transaction;
```

121 %

Messages

```
Msg 1205, Level 13, State 45, Line 3
Transaction (Process ID 64) was deadlocked on lock resources with another process and has been chosen as the deadlock victim. Rerun the transaction.

Completion time: 2021-11-07T17:01:43.8891054+05:30
```

SQLQuery2.sql - (l...RAGATI SINHA (64))* | SQLQuery1.sql - (l...RAGATI SINHA (62))*

```
begin transaction

update Book_Details set Book = 'Self comes to my mind' where Book_ID = 'Da001_Sel';

rollback transaction
commit transaction;


select * from Book_Details;
```

121 %

Messages

```
(1 row affected)

Completion time: 2021-11-07T17:01:43.9290488+05:30
```

Demo link : Demo

a) Justify that the situation is indeed one of a deadlock

The first transaction 1st update a row with book_ID = Mi009_Emo which locks the book_detail table .
So now in second transaction the same table is not able to update and query keeps on running.
Now when the 1st row also tries to update the same tuple which 2nd transaction is trying to update , 2nd transaction has locked it.
Hence situation of deadlock arise.
One of the transaction is chosen as victim and rolled back another one gets executed.

b. If you had to choose a victim to resolve the deadlock, what would your algorithm be?

There are a few victim choosing algorithm which can be used.

1) The transaction in which least no. of changes have been made so far in the database. As the victim transaction needs to rollback all the changes made . Hence it will be time effective work
2) Choosing the younger or older transaction as victim (wait-die or wound wait).
3) Change the default priority in ssms for any transaction.
   Reference link : SET DEADLOCK_PRIORITY | SSMS

C. Attempt execution of your algorithm for deadlock avoidance

Deadlock avoidance algo

d. Comment on the conflict serializability of your algorithm

For the deadlock on same tuple in part a as well as deadlock for 2 tables in part c we have performed Write , write operation. So both locks are X-locks.This implies reordering of statements will not help and hence it is not conflict serializable.

4. Simulate a deadlock situation derived from multiple transactions attempting to access & update the same table.

Demo

a) Justify that the situation is indeed one of a deadlock

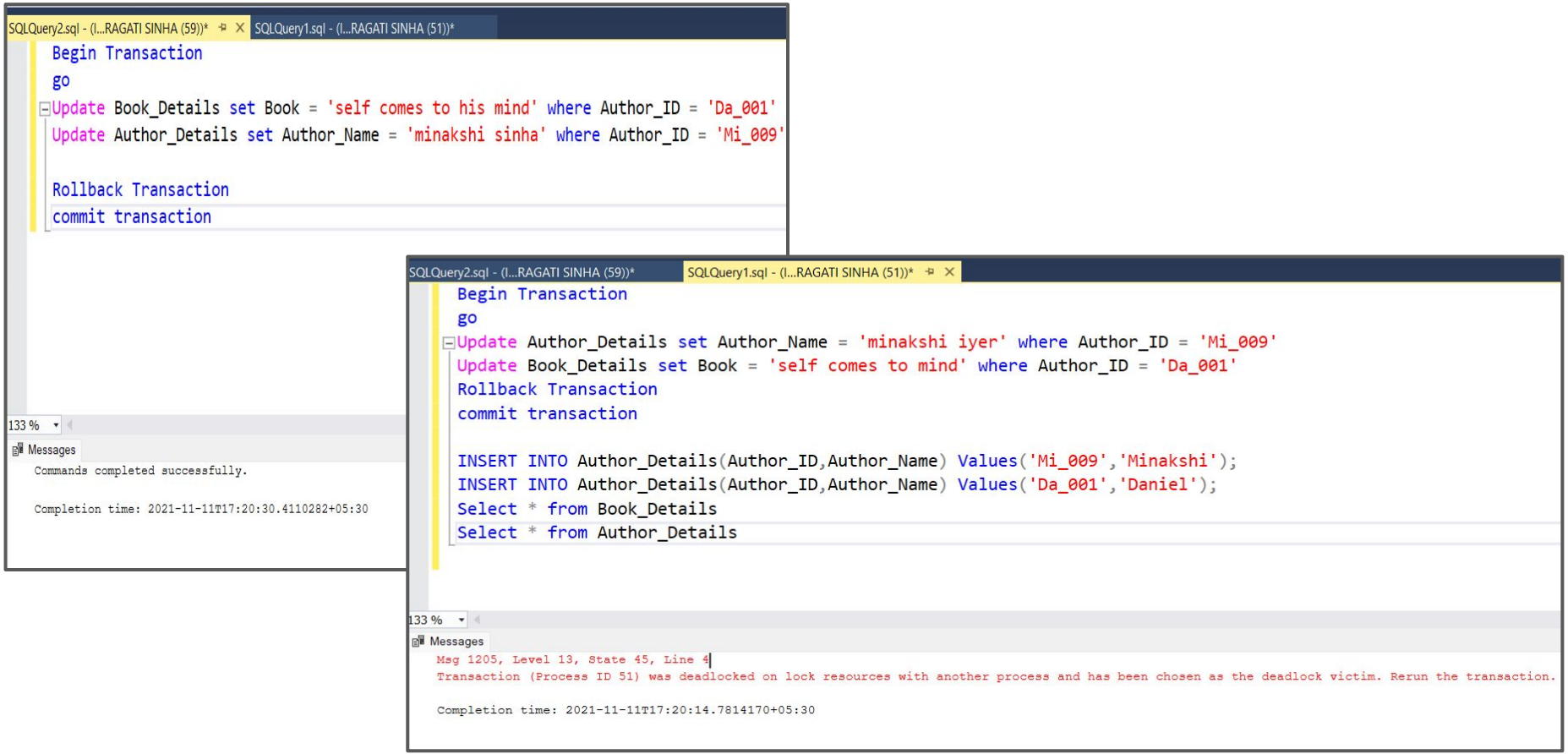
In the 1st transaction first Book Details table needs to be updated then author details table.

In the 2nd transaction first author detail table needs to be updated then book details table.

So 1st transaction locks Book_details table after executing 1st query.

2nd transaction locks author_details table after executing 1st query.

Now both the transaction won't be able to execute 2nd query . Hence there will be deadlock.



B. If you had to choose a victim to resolve the deadlock, what would your algorithm be?

There are a few victim choosing algorithm which can be used.

1) The transaction in which least no. of changes have been made so far in the database.

As the victim transaction needs to rollback all the changes made . Hence it will be time effective work. However here both transactions have equally expensive statement , so anyone can be chosen as victim.

2) Choosing the younger or older transaction as victim (wait-die or wound wait).
3) Change the default priority in ssms for any transaction.
   Reference link : SET DEADLOCK_PRIORITY | SSMS

C. Attempt execution of your algorithm for deadlock avoidance:

Deadlock avoidance algo

D.Comment on the conflict serializability of your algorithm

For the deadlock on 2 tables in part a as well as deadlock for 2 tables in part c we have performed Write , write operation. So both locks are X-locks.This implies reordering of statements will not help and hence it is not conflict serializable.

5. Create a real-life scenario in which an in-house application returns an error to the users, and users notify the development team about this error. The development team realizes that it is a deadlock issue, but they could not find the main reason for the problem. Under these circumstances, the team decides to receive consultancy service from an experienced database administrator. Do simulation for the same.

Output

DEMO

```
Transaction 0 has started
Transaction 1 has started
transaction 1 is has updated the first table 1
transaction 0 is has updated the first table 0
Both tables are locked.. User report a DEADLOCK
 Development  team decides to receive consultancy service from an experienced database administrator
Database administrator looks at locks and decides to unlock one of them by a rollback
<<<<<<      ROLLBACK A TRANSACTION    >>>>>>

Transaction 1 rolled back (releases 1 table)
transaction 0 is has updated the second table 1
transaction 0 has  completedsandy@LAPTOP-BG8QH1CU:/mnt/c/Users/sandi/OneDrive/Documents/Coding/DSA$
```

6. Take the following scenario & do the simulation for the same. Suppose we have two database sessions called A and B. Let's say that session A requests and has a lock on some data – and let's call the data Y. And then session B has a lock on some data that we will call Z. But now, let's say that session A needs a lock on data Z in order to run another SQL statement, but that lock is currently held by session B. And, let's say that session B needs a lock on data Y, but that lock is currently held by session A. This means that session B is waiting on session A's lock and session B is waiting for session A's lock.

This scenario is same to the scenario discussed in q4.

Demo

So in demo video there are two tables and two transactions happening. So A,B are two different transactions.

Transaction A locks book details table after executing first query . So table book details is data Y.

Transaction B locks author details table after executing first query . So table author details is data Z.

Now Transaction A wants to access author details i.e. data Z.

Now Transaction B wants to access book details i.e. data Y.

So deadlock arises and is automatically solved by SSMS software.

7. Simulate a deadlock situation to prove that The deadlock state can be changed back to stable state by using Rollback statement.

Demo: demo

So what happens on roll back is when that particular transaction is not allowing other transaction to access the table/tuple or item and has locked it , itself is not able to move further because its item is locked then it gets rolled back i.e. every statement executed in that transaction is removed and item is released.

8) Simulate the Wait-die situation in deadlock for When transaction Ti requests a data item currently held by Tj, Ti is allowed to wait only if it has a timestamp smaller than that of Tj (that is, Ti is older than Tj). Otherwise, Ti is rolled back (dies).

```
sandy@LAPTOP-BG8QH1CU:/mnt/c/Users/sandi/OneDrive/Documents/Coding/DSA$ ./w

Transaction 0 has started
Transaction 1 has started
transaction 1 is has updated the first table 1
transaction 0 is has updated the first table 0
 timestamp of transaction 1 is greater than 0 so ROLLBACK!!!
Transaction 1 rolled back (releases 1st table)
transaction 0 is has updated the second table 1
transaction 0 has  completedsandy@LAPTOP-BG8QH1CU:/mnt/c/Users/sandi/OneDrive/Documents/Coding/DSA$
```

Demo: DEMO