Lab 4

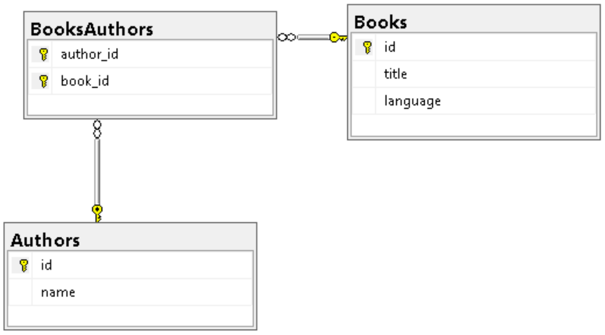
The first problem has to be solved in SQL SERVER. The last problem has to be solved in C#.

- Creaţi 4 scenarii ce reproduc următoarele situaţii generate de execuţia concurentă: *dirty reads, non-repeatable reads, phantom reads* şi un *deadlock*. Puteţi implementa aceste scenarii atât ca proceduri stocate cât şi ca interogări de sine stătătoare. De asemenea, pentru fiecare dintre scenariile create, găsiţi soluţii de rezolvare/evitare a acestor situaţii. (nota: 7)

You need to consider a table in which you will analyze the concurrency execution. Here I choose Books. You must prepare scenarios for each case: (Transaction 1 with Transaction 2) and (Transaction 1 with Transaction 2) 'solved'. You have to create and save each of the transactions used. You can use one file for Transaction 1 and one file for Transaction 2, with both of the cases (unsolved and solved- also commented), or 2 files, saved suggestive. Or, you can organize the structure as you prefer, but to be clear. Also, prepare examples for each of the cases.

Try to run the transactions in the same time (or close). Start Transaction 1 first, introduce a delay there, so that Transaction 2 can be executed in that time. Immediately that Transaction 1 was started, start also Transaction 2. (If you run the transactions converse, the result will also be converse).

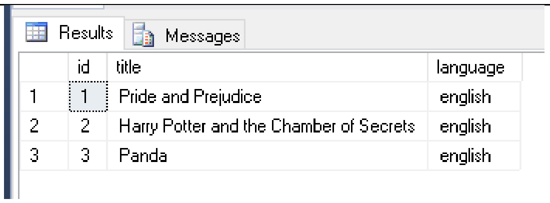
We consider the database



```
create database DBMS_Lab3_
go
use DBMS_Lab3_
go

CREATE TABLE Books(
id INT PRIMARY KEY IDENTITY,
title varchar(100),
language varchar(100))

CREATE TABLE Authors(
id INT PRIMARY KEY IDENTITY,
name varchar(100))

CREATE TABLE BooksAuthors(
author_id INT FOREIGN KEY REFERENCES Authors(id),
book_id INT FOREIGN KEY REFERENCES Books(id),
CONSTRAINT pk_BooksAuthors PRIMARY KEY (author_id, book_id))
```

In table Books we have

```
-- add records
insert into Books values
('Pride and Prejudice', 'english'),
('Harry Potter and the Chamber of Secrets',
'english'), ('Panda', 'english')

select * from Books
```



	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english

Please, put in each file `use DBMS_Lab3_`

In what follows, we will work with the table Books and

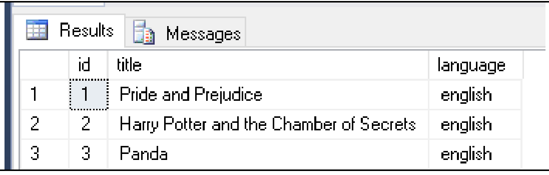
T1=Transaction 1 starts first and finish first

T2=Transaction start immediately after T1 and finish after T1

DIRTY READS – T1: update + delay + rollback, T2: select + delay + select -> see the update in the first select, even if it is rollback then (the order in the execution of the operations is: update – select – rollback – select)

Isolation level: Read Uncommitted / Read Committed (solution)

```
-- dirty reads
select * from Books
```

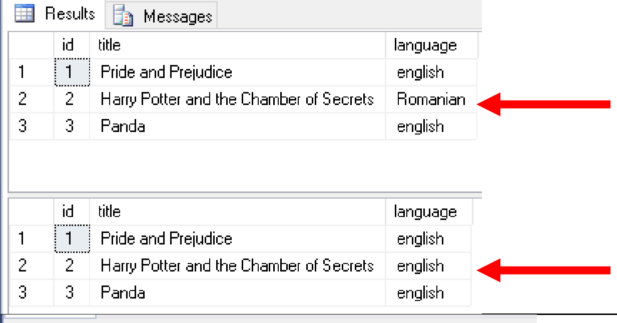


	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english

```
BEGIN TRANSACTION
UPDATE Books SET
language='Romanian' WHERE id = 2
WAITFOR DELAY '00:00:10'
ROLLBACK TRANSACTION
```

```
--PROBLEM: SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
BEGIN TRAN
SELECT * FROM Books
WAITFOR DELAY '00:00:15'
SELECT * FROM Books
COMMIT TRAN
```

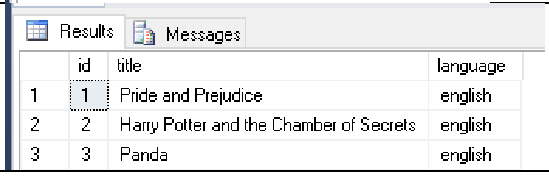
```
(1 row(s) affected)
```



	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	Romanian
3	3	Panda	english

	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english

```
-- dirty reads
select * from Books
```

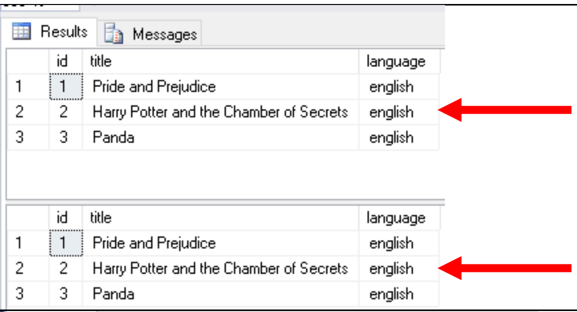


	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english

```
BEGIN TRANSACTION
UPDATE Books SET
language='Romanian' WHERE id = 2
WAITFOR DELAY '00:00:10'
ROLLBACK TRANSACTION
```

```
-- SOLUTION: SET TRANSACTION ISOLATION LEVEL TO READ COMMITTED
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
BEGIN TRAN
SELECT * FROM Books
WAITFOR DELAY '00:00:15'
SELECT * FROM Books
COMMIT TRAN
```

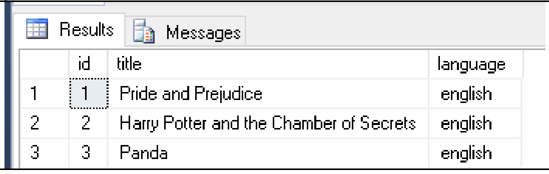
```
(1 row(s) affected)
```



	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english

	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english

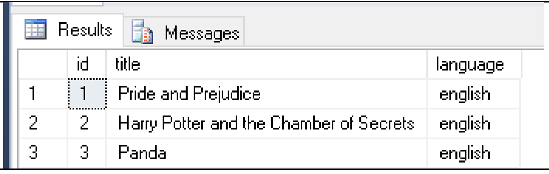
```
-- dirty reads
select * from Books
```



	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english

1. NON-REPEATABLE READS – T1: delay + update + commit, T2: select + delay + select -> see the value inserted before the transaction from the first select of T2 + see the update of the value inserted before the transaction, from the second select of T2 (the order in the execution of the operations is: select – update – select)
   Isolation level: Read Committed / Repeatable Read (solution)

```
-- nonrepeatable reads
select * from Books
```

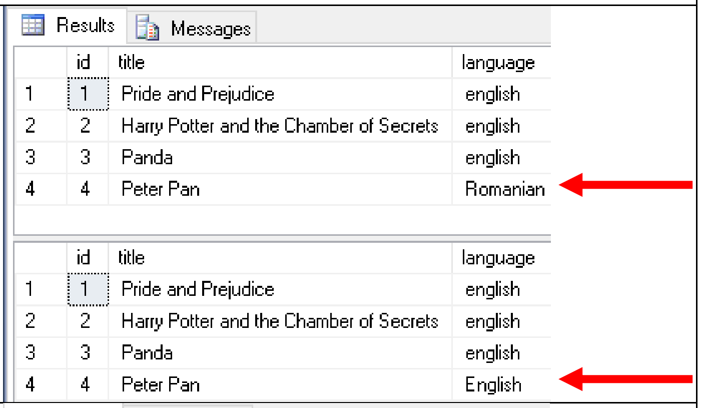


	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english

```
INSERT INTO Books(title, language) VALUES ('Peter
Pan','Romanian')
BEGIN TRAN
WAITFOR DELAY '00:00:10'
UPDATE Books SET language='English' WHERE title =
'Peter Pan'
COMMIT TRAN
```

```
--PROBLEM: SET TRANSACTION ISOLATION LEVEL READ
UNCOMMITTED
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
BEGIN TRAN
SELECT * FROM Books
WAITFOR DELAY '00:00:15'
SELECT * FROM Books
COMMIT TRAN
```

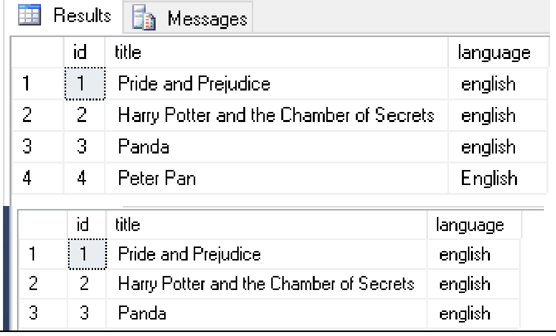
```
(1 row(s) affected) -- from insert

(1 row(s) affected) – from update
```



	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english
4	4	Peter Pan	Romanian

	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english
4	4	Peter Pan	English

```
-- nonrepeatable reads
select * from Books
delete from Books where title = 'Peter Pan'
select * from Books
```



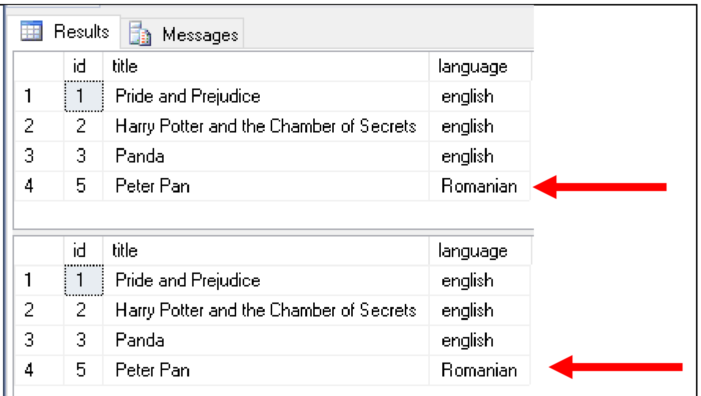
	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english
4	4	Peter Pan	English

	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english

```
INSERT INTO Books(title, language) VALUES ('Peter
Pan','Romanian')
BEGIN TRAN
WAITFOR DELAY '00:00:10'
UPDATE Books SET language='English' WHERE title =
'Peter Pan'
COMMIT TRAN
```

```
-- SOLUTION: SET TRANSACTION ISOLATION LEVEL TO
REPEATABLE READ
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
BEGIN TRAN
SELECT * FROM Books
WAITFOR DELAY '00:00:15'
SELECT * FROM Books
COMMIT TRAN
```

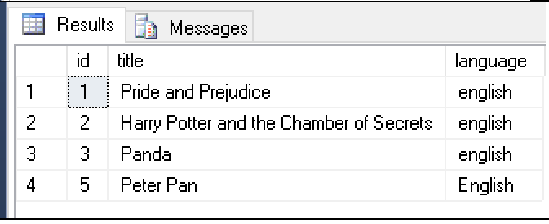
```
(1 row(s) affected)

(1 row(s) affected)
```



	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english
4	5	Peter Pan	Romanian

	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english
4	5	Peter Pan	Romanian

```
-- nonrepeatable reads
select * from Books
```

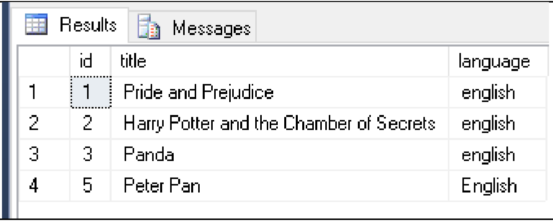


	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english
4	5	Peter Pan	English

2. PHANTOM READS – T1: delay + insert + commit, T2: select + delay + select -> see the inserted value only at the second select from T2 (the order in the execution of the operations is: select – insert – select)

Isolation level: Repeatable Read / Serializable (solution)

```
-- phantom reads
select * from Books
```

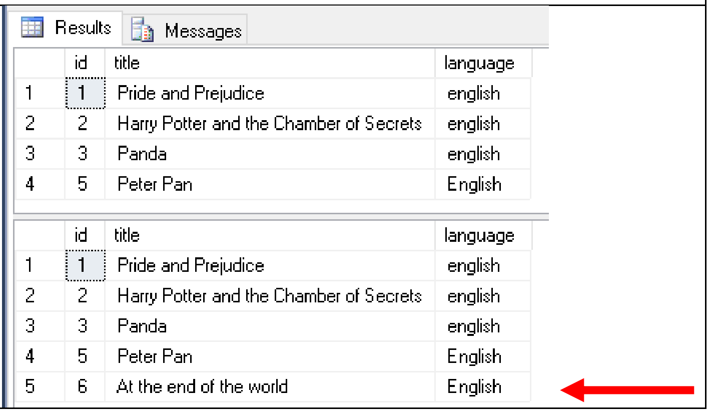


	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english
4	5	Peter Pan	English

```
BEGIN TRAN
WAITFOR DELAY '00:00:10'
INSERT INTO Books(title,language) VALUES ('At
the end of the world','English')
COMMIT TRAN
```

```
-- PROBLEM: SET TRANSACTION ISOLATION LEVEL TO
REPEATABLE READ
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
BEGIN TRAN
SELECT * FROM Books
WAITFOR DELAY '00:00:15'
SELECT * FROM Books
COMMIT TRAN
```

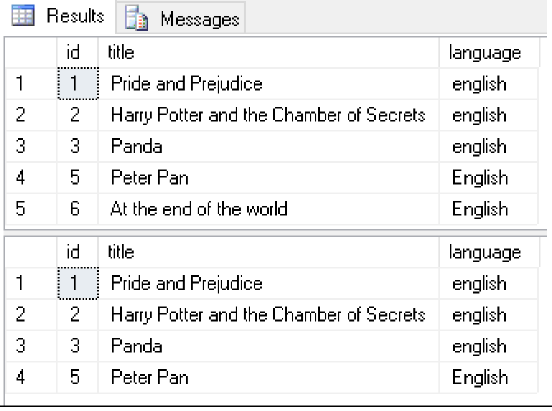
```
(1 row(s) affected)
```



	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english
4	5	Peter Pan	English

	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english
4	5	Peter Pan	English
5	6	At the end of the world	English

```
-- phantom reads
select * from Books
delete from Books where title='At the end of
the world'
select * from Books
```



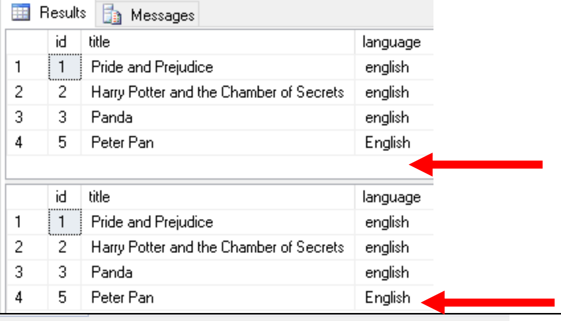
	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english
4	5	Peter Pan	English
5	6	At the end of the world	English

	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english
4	5	Peter Pan	English

```
BEGIN TRAN
WAITFOR DELAY '00:00:10'
INSERT INTO Books(title,language) VALUES ('At
the end of the world','English')
COMMIT TRAN
```

```
-- SOLUTION: SET TRANSACTION ISOLATION LEVEL TO
SERIALIZABLE
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
BEGIN TRAN
SELECT * FROM Books
WAITFOR DELAY '00:00:15'
SELECT * FROM Books
COMMIT TRAN
```

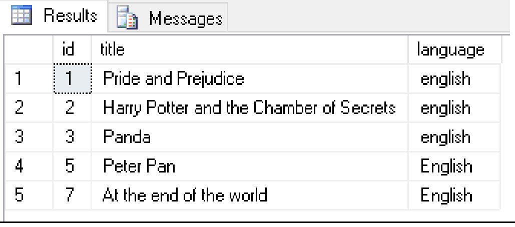
```
(1 row(s) affected)
```



	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english
4	5	Peter Pan	English

	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english
4	5	Peter Pan	English

```
-- phantom reads
select * from Books
```



	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english
4	5	Peter Pan	English
5	7	At the end of the world	English

DEADLOCK - T1: update on table A + delay + update on table B

- T2: update on table B + delay + update on table A

T1- update on table A -> exclusive lock on table A	delay	Try to update table B	Table B is blocked in T2	One of the blocked transactions, T1 or T2, will be chosen as a deadlock victim and terminates with an error. The other transaction wins and update both table A and table B
T2 - update on table B -> exclusive lock on table B	delay	Try to update table A	Table A is blocked in T1	

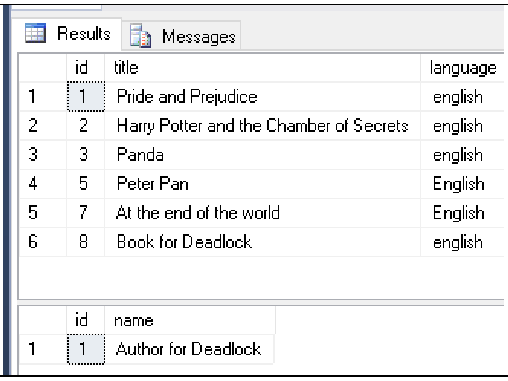
The only solution is to decide which of the 2 transactions to win, by using the DEADLOCK_PRIORITY, that can be set (LOW, NORMAL, HIGH, or from -10 (-5) to 10 (5)). Implicit is NORMAL (0).

The victim transaction is chosen like this:

1. The transaction with the lowest DEADLOCK_PRIORITY
2. If both of the transactions have the same DEADLOCK_PRIORITY, the victim is the one, less expensive at ROLLBACK
3. If both of the transactions have the same DEADLOCK_PRIORITY and the same cost, the victim is chosen randomly

We consider tables: Books and Authors

```sql
-- deadlock
insert into Books values ('Book for Deadlock',
'english')
insert into Authors values ('Author for Deadlock')
select * from Books
select * from Authors
```



	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english
4	5	Peter Pan	English
5	7	At the end of the world	English
6	8	Book for Deadlock	english

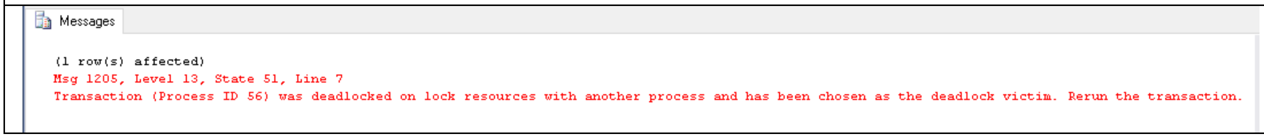
	id	name
1	1	Author for Deadlock

Deadlock example:

```sql
-- transaction 1
begin tran
update Books set title='deadlock Books Transaction 1' where id=8
-- this transaction has exclusively lock on table Books
waitfor delay '00:00:10'
update Authors set name='deadlock Authors Transaction 1' where id=1
commit tran
```

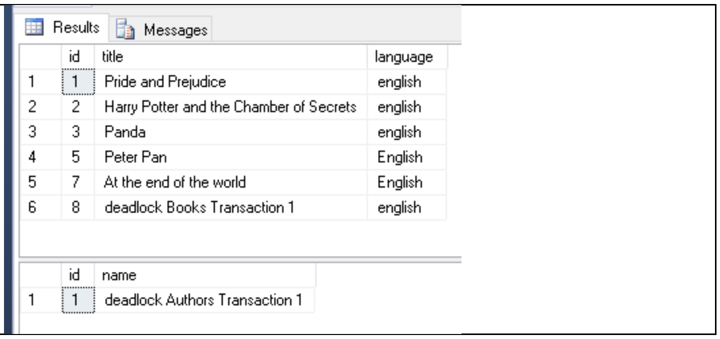
```
(1 row(s) affected)

(1 row(s) affected)
```

```sql
-- transaction 2
begin tran
update Authors set name='deadlock Authors Transaction 2' where id=1
-- this transaction has exclusively lock on table Authors
waitfor delay '00:00:10'
update Books set title='deadlock Books Transaction 2' where id=8
commit tran
```



```
(1 row(s) affected)
Msg 1205, Level 13, State 51, Line 7
Transaction (Process ID 56) was deadlocked on lock resources with another process and has been chosen as the deadlock victim. Rerun the transaction.
```

```
select * from Books
select * from Authors


So, transaction 1 is the winner and update the
tables Books and Authors, and transaction 2 is the
victim
```

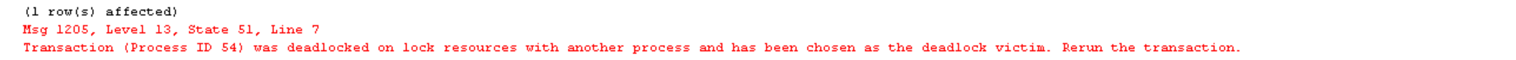


	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english
4	5	Peter Pan	English
5	7	At the end of the world	English
6	8	deadlock Books Transaction 1	english

	id	name
1	1	deadlock Authors Transaction 1

If in transaction 2, we set DEADLOCK_PRIORITY to HIGH, or, if in transaction 1 we set DEADLOCK_PRIORITY to LOW, the winner will be transaction 2 and the victim transaction 1.

```
-- transaction 1
begin tran
update Books set title='deadlock Books Transaction 1' where id=8
-- this transaction has exclusively lock on table Books
waitfor delay '00:00:10'
update Authors set name='deadlock Authors Transaction 1' where id=1
commit tran
```

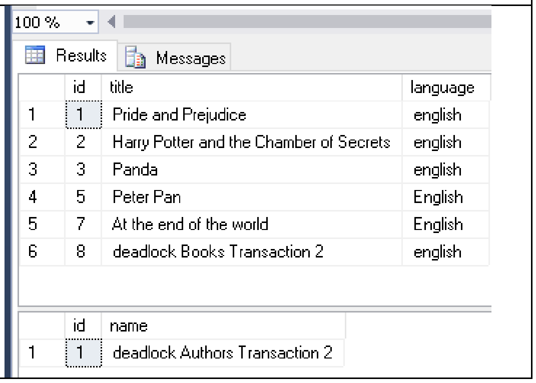


```
(1 row(s) affected)
Msg 1205, Level 13, State 51, Line 7
Transaction (Process ID 54) was deadlocked on lock resources with another process and has been chosen as the deadlock victim. Rerun the transaction.
```

```
-- transaction 2
SET DEADLOCK_PRIORITY HIGH
-- SET DEADLOCK_PRIORITY LOW
begin tran
update Authors set name='deadlock Authors Transaction 2' where id=1
-- this transaction has exclusively lock on table Authors
waitfor delay '00:00:10'
update Books set title='deadlock Books Transaction 2' where id=8
commit tran
```

```
(1 row(s) affected)

(1 row(s) affected)
```

```
select * from Books
select * from Authors
```



	id	title	language
1	1	Pride and Prejudice	english
2	2	Harry Potter and the Chamber of Secrets	english
3	3	Panda	english
4	5	Peter Pan	English
5	7	At the end of the world	English
6	8	deadlock Books Transaction 2	english

	id	name
1	1	deadlock Authors Transaction 2

- Creaţi un scenariu de *deadlock* prin intermediul unei aplicaţii .NET, folosind *multithreading*. Va trebui ca două proceduri stocate/interogări să fie executate în 2 fire de execuţie diferite. Firul de execuţie ce eşuează din cauza *deadlock*-ului va trebui să fie reluat (stabiliţi un număr maxim de reluări până când procedura stocată/interogarea este considerată terminată fără succes - *aborted*). (nota: 10)

There are 2 possibilities:

- create the stored procedures in SQL Server and only use them in C# for 2 threads with locks

- create everything in C#.

***Observaţie**: Pentru toate scenariile trebuie să stabiliţi un sistem de logare ce vă va permite să memoraţi istoricul acţiunilor executate. Pentru detectarea erorilor se recomandă folosirea clauzei* try-catch*, atât în aplicaţia .NET cât şi în codul SQL.*

It is up to you, how you decide to have the log system, but please explain.

Good luck ☺