Lab 4

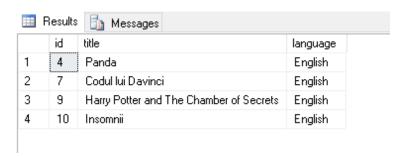
The first problem has to be solved in SQL SERVER. The last problem has to be solved in C#.

- Creați 4 scenarii ce reproduc următoarele situații generate de execuția concurentă: *dirty reads*, *non-repeatable reads*, *phantom reads* și un *deadlock*. Puteți implementa aceste scenarii atât ca proceduri stocate cât și ca interogări de sine stătătoare. De asemenea, pentru fiecare dintre scenariile create, găsiți soluții de rezolvare/evitare a acestor situații. (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).

In table Books we have



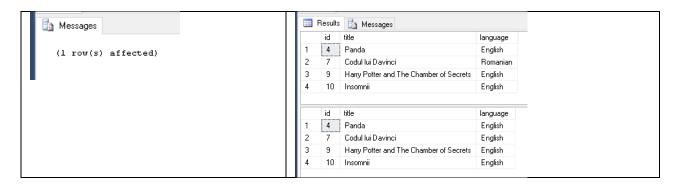
For what follows: T1=Transaction 1 starts first. T2=Transaction start immediately after T1.

1. DIRTY READS – T1: 1 update + delay + rollback, T2: select + delay + select -> we see the update in the first select (T1 – finish first), even if it is rollback then Isolation level: Read Uncommitted / Read Committed (solution)

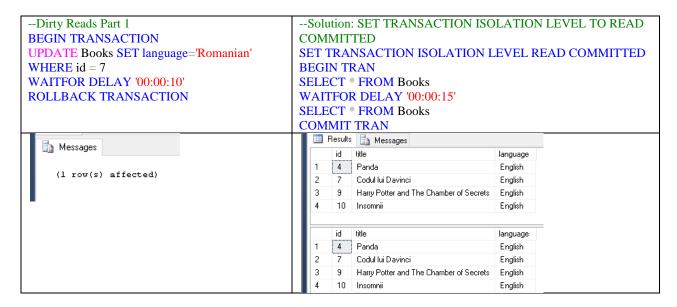
--Dirty Reads Part 1
BEGIN TRANSACTION
UPDATE Books SET language='Romanian'
WHERE id = 7
WAITFOR DELAY '00:00:10'
ROLLBACK TRANSACTION

WAITFOR DELAY '00:00:10'
ROLLBACK TRANSACTION

--Dirty Reads Part 2
SET TRANSACTION ISOLATION LEVEL READ
UNCOMMITTED
BEGIN TRAN
SELECT * FROM Books
WAITFOR DELAY '00:00:15'
SELECT * FROM Books
COMMIT TRAN

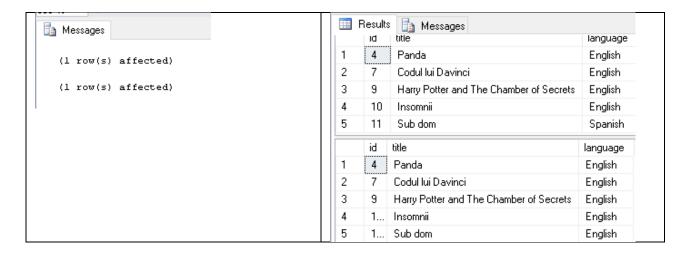


Solution: T1: 1 update + delay + rollback, T2: select + delay + select -> we don't see the update (that is also rollback) – T1 finish first

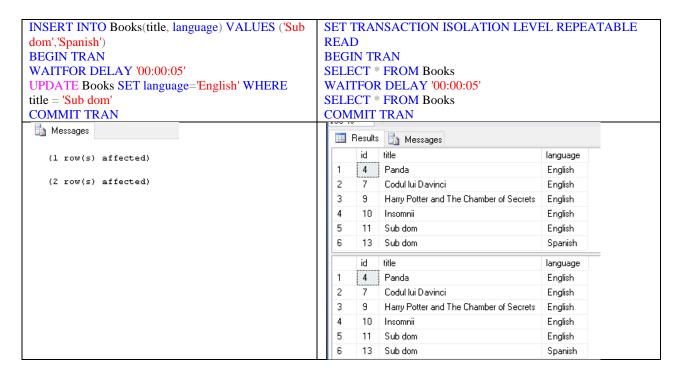


2. NON-REPEATABLE READS – T1: insert + delay + update + commit, T2: select + delay + select -> see the insert in first select of T2 + update in the second select of T2, T1 finish first Isolation level: Read Committed / Repeatable Read (solution). The result will contain the previous row version (before the finish of the transaction).

INSERT INTO Books(title, language) VALUES ('Sub	SET TRANSACTION ISOLATION LEVEL READ			
dom','Spanish')	COMMITTED			
BEGIN TRAN	BEGIN TRAN			
WAITFOR DELAY '00:00:05'	SELECT * FROM Books			
UPDATE Books SET language='English' WHERE	WAITFOR DELAY '00:00:05'			
title = 'Sub dom'	SELECT * FROM Books			
COMMIT TRAN	COMMIT TRAN			



Solution: T1: insert + delay + update + commit, T2: select + delay + select -> see only the final result in both of the select of T2, T1 finish first

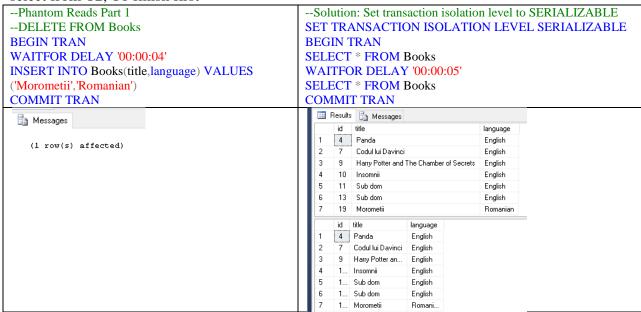


PHANTOM READS – T1: delay + insert + commit, T2: select + delay + select -> see the inserted value only at the second select from T2, T1 finish first. The result will contain the previous row version; the same number of rows (before the finish of the transaction – for example, 5 not 6). Isolation level: Repeatable Read / Serializable (solution)

Phantom Reads Part 1	Phantom Reads Part 2				
DELETE FROM Books	SET TRANSACTION ISOLATION LEVEL REPEATABLE				
BEGIN TRAN	READ				
WAITFOR DELAY '00:00:04'	BEGIN TRAN				
INSERT INTO Books(title,language) VALUES	SELECT * FROM Books				
('Morometii', 'Romanian')	WAITFOR DELAY '00:00:05'				
COMMIT TRAN	SELECT * FROM Books				

	COMMIT TRAN							
Messages				Results 🔒 Messages				
(1 row(s) affected)			id	title	language			
		1	4	Panda	English			
		2	7	Codul lui Davinci	English			
		3	9	Harry Potter and The Chamber of Secrets	English			
		4	10	Insomnii	English			
		5	11	Sub dom	English			
		6	13	Sub dom	English			
			id	title	language	±		
		1	4	Panda	English			
		2	7	Codul lui Davinci	English			
		3	9	Harry Potter and The Chamber of Secrets	English			
		4	1	Insomnii	English			
		5	1	Sub dom	English			
		6	1	Sub dom	English			
		7	1	Morometii	Romani			

Solution: T1: delay + insert + commit, T2: select + delay + select -> see the inserted value in both of the select from T2, T1 finish first

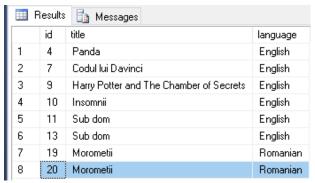


3. DEADLOCK – T1: update on table A + delay + update on table B, T2: update on table B + delay + update on table A

We update on table A (from T1 – that exclusively lock on table A), update on table B (from T2 – that exclusively lock on table B), try to update from T1 table B (but this transaction will be blocked because T2 has already been locked on table B), try to update from T2 table A (but this transaction will be blocked because T1 has already been locked on table A). So, both of the transactions are blocked. After some seconds T2 will be chosen as a deadlock victim and terminates with an error. After that, T1 will finish also. In table A and table B will be the values from T1.

Here we consider 2 tables: Books, Authors. Books

Authors



5

6

7

8

11

13

19

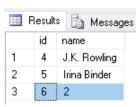
20

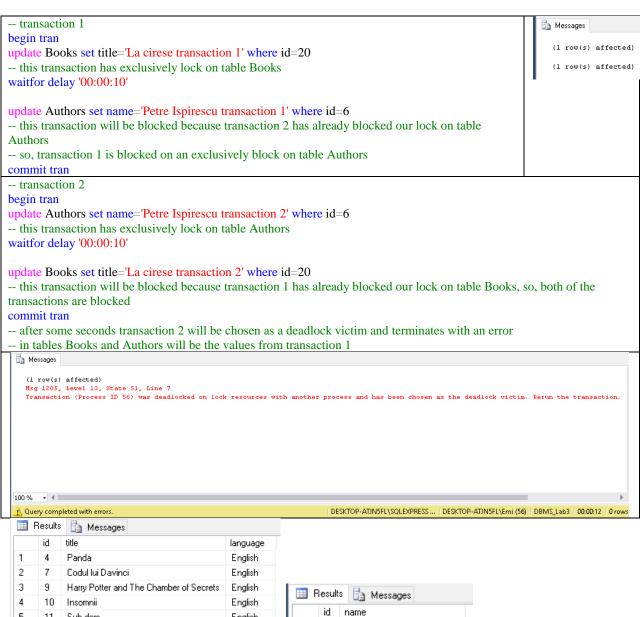
Sub dom

Sub dom

Morometii

La cirese transaction 1





4

5

6

2

3

J.K. Rowling

Irina Binder

Petre Ispirescu transaction 1

English

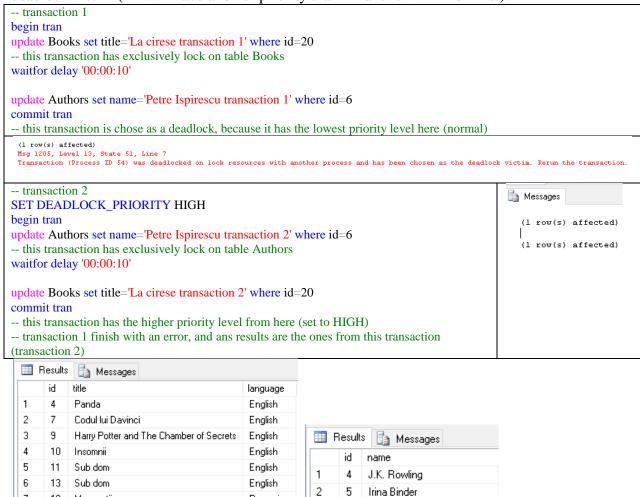
English

Romanian

Romanian

Solution: For deadlock, the priority has to be set (LOW, NORMAL, HIGH, or from -10 to 10). Implicit is NORMAL (0).

For example, here we set the DEADLOCK_PRIORITY to HIGH for T2, so that T1 be chosen as a deadlock victim (T1 will have a lower priority than T2 and it will finish first).



Romanian

Romanian

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

Petre Ispirescu transaction 2

There are 2 possibilities:

7

19

20

Morometii

La cirese transaction 2

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

Observație: Pentru toate scenariile trebuie să stabiliți un sistem de logare ce vă va permite să memorați istoricul acțiunilor executate. Pentru detectarea erorilor se recomandă folosirea clauzei try-catch, atât în aplicația .NET cât și în codul SQL.

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

Good luck ©