SE 308 Term Project 1

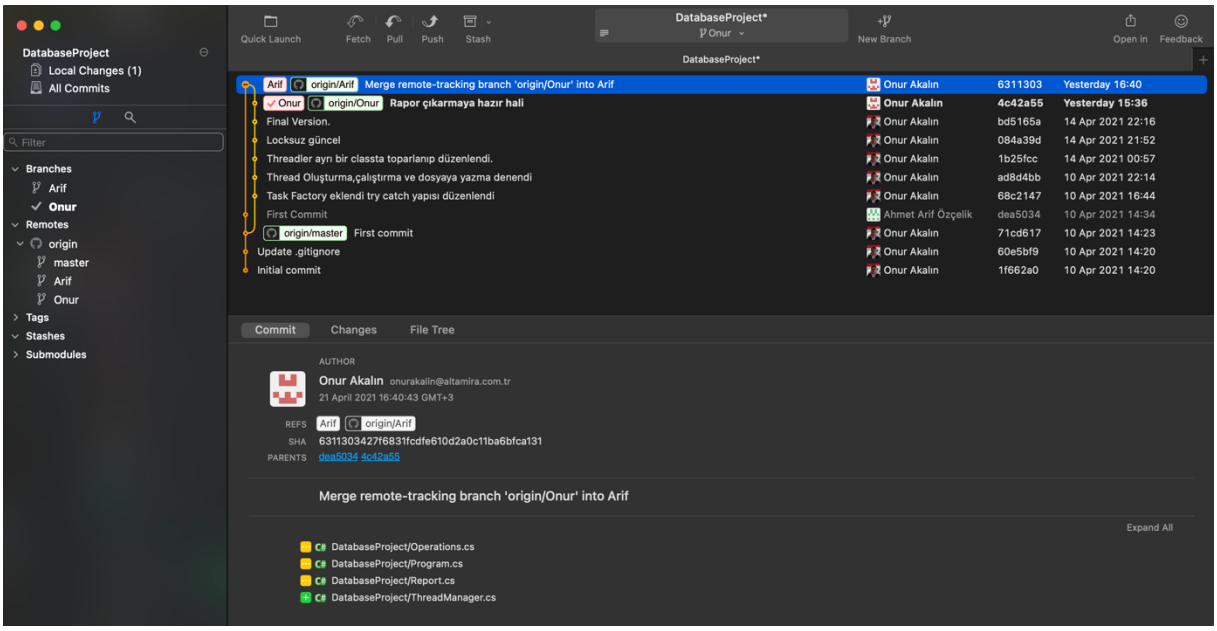
Onur Akalın 170706018 & Ahmet Arif Özçelik 170706016

Introduction

We worked 2 people in this simulation experiment, so we got different results from different computers. In this way, we think our results are more accurate. We developed our program using .Net Core, ADO.Net and MSSQL. The reason we use these environments is that we have previous experience with them and they provide us convenience in matters such as threads, transactions and sql.

Project

While developing the program, we used git as a version control tool. Since we are working in different branches, there are many versions of the project that we can test. In general, we used the pair programming technique. Having all our versions on GitHub repository has given us great advantages. While testing some new and inexperienced developments, we were able to return to the old versions with one click, although we completely broke the code.



After completing our thread structure, we found that this thread took 15 minutes for 1 user. Therefore, we started to debug the entire thread structure together. As a result of our debug operations, we caught a very interesting exception in our try catch structure. This exception showed us a trigger fail error message, the message said that the limit of a column in the table was exceeded. On top of that, when we checked the database, we saw 2 triggers that affect the queries which we use in our project. After that, we thought that this issue would affect everyone and we shared it with our teacher in the lesson and we decided to delete the triggers completely. After the triggers were removed, our threads started to run 10 times faster.

We used nested try catch structures in the management of exceptions that occur during the running process of the program. The reason for using such a structure is to better categorize the exceptions that occur. As a result, we grouped the exceptions in two categories as deadlock and other types, so we did not interfere with each other and obtained a better and more accurate report.

We wrote a function in our project to prepare the requested report. This function saves the outputs generated after threads run in a text file, according to the number of users entered.

These outputs are:

- TypeA User Count
- TypeA Deadlock Count
- TypeB Deadlock Count
- TypeA Average Time Cost
- TypeB Average Time Cost
- Other Exception Count

Samples of .txt files that we generated

ReadUncommitted.txt

```
*************************************************************
TypeA User Count : 1 TypeB User Count : 1
TypeA Deadlock Count : 0
TypeB Deadlock Count : 0
TypeA Average Time Cost : 00:00:14.5963750
TypeB Average Time Cost : 00:00:07.8359970
Other Exception Count : 0
*************************************************************

*************************************************************
TypeA User Count : 2 TypeB User Count : 2
TypeA Deadlock Count : 85
TypeB Deadlock Count : 0
TypeA Average Time Cost : 00:00:41.7329890
TypeB Average Time Cost : 00:00:05.7240480
Other Exception Count : 0
*************************************************************

*************************************************************
TypeA User Count : 3 TypeB User Count : 3
TypeA Deadlock Count : 187
TypeB Deadlock Count : 0
TypeA Average Time Cost : 00:00:52.3372470
TypeB Average Time Cost : 00:00:05.6202190
Other Exception Count : 0
*************************************************************

*************************************************************
TypeA User Count : 4 TypeB User Count : 4
TypeA Deadlock Count : 288
TypeB Deadlock Count : 0
TypeA Average Time Cost : 00:00:55.8887920
TypeB Average Time Cost : 00:00:08.2160595
Other Exception Count : 0
*************************************************************

*************************************************************
TypeA User Count : 5 TypeB User Count : 5
TypeA Deadlock Count : 366
TypeB Deadlock Count : 0
TypeA Average Time Cost : 00:01:13.7459560
TypeB Average Time Cost : 00:00:10.7579744
Other Exception Count : 0
*************************************************************

*************************************************************
TypeA User Count : 10 TypeB User Count : 10
TypeA Deadlock Count : 841
TypeB Deadlock Count : 0
TypeA Average Time Cost : 00:02:03.0266214
TypeB Average Time Cost : 00:00:27.0669890
Other Exception Count : 0
```

Serializable.txt

```
*************************************************************
TypeA User Count : 1 TypeB User Count : 1
TypeA Deadlock Count : 0
TypeB Deadlock Count : 1
TypeA Average Time Cost : 00:00:34.3639360
TypeB Average Time Cost : 00:00:29.8813030
Other Exception Count : 0
*************************************************************

*************************************************************
TypeA User Count : 2 TypeB User Count : 2
TypeA Deadlock Count : 14
TypeB Deadlock Count : 4
TypeA Average Time Cost : 00:01:30.4106445
TypeB Average Time Cost : 00:01:01.8513360
Other Exception Count : 0
*************************************************************

*************************************************************
TypeA User Count : 3 TypeB User Count : 3
TypeA Deadlock Count : 124
TypeB Deadlock Count : 12
TypeA Average Time Cost : 00:02:22.3159220
TypeB Average Time Cost : 00:01:14.2875697
Other Exception Count : 0
*************************************************************

*************************************************************
TypeA User Count : 4 TypeB User Count : 4
TypeA Deadlock Count : 197
TypeB Deadlock Count : 9
TypeA Average Time Cost : 00:02:42.2282868
TypeB Average Time Cost : 00:00:52.3455900
Other Exception Count : 0
*************************************************************

*************************************************************
TypeA User Count : 5 TypeB User Count : 5
TypeA Deadlock Count : 289
TypeB Deadlock Count : 17
TypeA Average Time Cost : 00:03:10.2675016
TypeB Average Time Cost : 00:01:11.9858802
Other Exception Count : 0
*************************************************************

*************************************************************
TypeA User Count : 10 TypeB User Count : 10
TypeA Deadlock Count : 686
TypeB Deadlock Count : 24
TypeA Average Time Cost : 00:05:09.8762694
TypeB Average Time Cost : 00:01:23.4352911
Other Exception Count : 0
```

Report

We created the necessary tables by examining the txt files, which are the output of our program. In order to share the workload and see different results, we divided the isolation levels into 2 parts.

Onur Akalın: Read Uncommitted and Serializable

Ahmet Arif Özçelik: Read Committed and Repetable Read

Average times were used in minutes and seconds. We increased the average deadlocks counters with thread lock so that we did not get deadlock while trying to increase counter between threads.

Isolation Level	READ UNCOMMITED – (Done by Onur)				
Number of Type A Users	**Number of Type B Users**	**Average Time of Type A Threads**	**Number of Deadlocks Encounterd by Type A Users**	**Average Time of Type B Threads**	**Number of Deadlocks Encounterd by Type B Users**
1	1	00:14	0	00:07	0
2	2	00:41	85	00:05	0
3	3	00:52	187	00:05	0
4	4	00:55	288	00:08	0
5	5	01:13	366	00:10	0
10	10	02:03	841	00:27	0
15	15	03:08	1328	00:32	0
25	25	05:36	2284	00:29	0
50	50	09:54	4697	01:39	0
100	100	22:23	9481	05:30	0

Read Uncommited

In this isolation level, we caught many deadlocks in the type A section. On the other hand, no deadlocks were caught in the type B section. As you can see, the number of deadlocks directly affects the average time. This isolation level shows clearly that it has no reducing effect on the number of deadlocks. At the same time, we can see that the deadlock numbers increase linearly in this section.

Isolation Level	READ COMMITED – (Done by Arif)				
Number of Type A Users	**Number of Type B Users**	**Average Time of Type A Threads**	**Number of Deadlocks Encounterd by Type A Users**	**Average Time of Type B Threads**	**Number of Deadlocks Encounterd by Type B Users**
1	1	00:25	0	00:27	0
2	2	00:54	17	00:49	0
3	3	01:38	69	00:37	0
4	4	02:48	134	01:17	0
5	5	02:49	183	01:28	0
10	10	05:06	642	01:46	0
15	15	07:31	1042	03:55	0
25	25	09:38	2014	05:41	0
50	50	17:23	4196	09:45	0
100	100	25:06	8845	27:10	0

Read Commited

Likewise in this isolation level, we caught a lot of deadlocks in the type A section and no deadlocks in the type B section. If We compare with Read Uncommited, It has more average time and less catched deadlock numbers. Again, we can see that the deadlock numbers increase linearly in this section.

Isolation Level	REPETABLE READ – (Done by Arif)				
Number of Type A Users	**Number of Type B Users**	**Average Time of Type A Threads**	**Number of Deadlocks Encounterd by Type A Users**	**Average Time of Type B Threads**	**Number of Deadlocks Encounterd by Type B Users**
1	1	00:41	0	00:41	1
2	2	01:43	11	01:23	2
3	3	03:23	36	02:30	3
4	4	03:43	52	02:09	24
5	5	04:57	91	02:43	24
10	10	08:44	261	03:05	147
15	15	11:35	428	05:03	126
25	25	15:35	1067	04:59	159
50	50	25:12	1878	09:21	206
100	100	45:12	1703	30:56	197

Repetable Read

Unlike the other isolation levels,we got different results at this level. Average time occurred as we expected. Although the number of deadlocks increases linearly at other isolation levels, there are some exceptions at this level as shown in the table.

Isolation Level	SERIALIZABLE – (Done by Onur)				
Number of Type A Users	**Number of Type B Users**	**Average Time of Type A Threads**	**Number of Deadlocks Encounterd by Type A Users**	**Average Time of Type B Threads**	**Number of Deadlocks Encounterd by Type B Users**
1	1	00:34	0	00:28	1
2	2	01:53	15	01:09	6
3	3	02:06	117	00:35	0
4	4	02:56	162	00:23	0
5	5	04:29	276	01:33	21
10	10	04:21	666	00:18	0
15	15	07:40	1244	03:48	37
25	25	24:43	1867	10:07	523
50	50	27:54	952	07:51	50
100	100	61:17	714	14:29	1

Serializable

This isolation level produced very different results compared to other levels. We could not get stable results, neither in average time nor number of deadlocks. These results made us suspicious and we decided to run our program again at this isolation level. Each time we run our program, we saw very interesting results. We wanted to include the outputs of these results in this report. Although the average times seemed short, we waited for a very long time when we experimented with 50 and 100 people in our study.

Serializable.txt

```
**************************************************************
TypeA User Count : 1 TypeB User Count : 1
TypeA Deadlock Count : 0
TypeB Deadlock Count : 1
TypeA Average Time Cost : 00:00:34.3639360
TypeB Average Time Cost : 00:00:29.8813030
Other Exception Count : 0
**************************************************************

**************************************************************
TypeA User Count : 2 TypeB User Count : 2
TypeA Deadlock Count : 14
TypeB Deadlock Count : 4
TypeA Average Time Cost : 00:01:30.4106445
TypeB Average Time Cost : 00:01:01.8513360
Other Exception Count : 0
**************************************************************

**************************************************************
TypeA User Count : 3 TypeB User Count : 3
TypeA Deadlock Count : 124
TypeB Deadlock Count : 12
TypeA Average Time Cost : 00:02:22.3159220
TypeB Average Time Cost : 00:01:14.2875697
Other Exception Count : 0
**************************************************************

**************************************************************
TypeA User Count : 4 TypeB User Count : 4
TypeA Deadlock Count : 197
TypeB Deadlock Count : 9
TypeA Average Time Cost : 00:02:42.2282868
TypeB Average Time Cost : 00:00:52.3455900
Other Exception Count : 0
**************************************************************

**************************************************************
TypeA User Count : 5 TypeB User Count : 5
TypeA Deadlock Count : 289
TypeB Deadlock Count : 17
TypeA Average Time Cost : 00:03:10.2675016
TypeB Average Time Cost : 00:01:11.9858802
Other Exception Count : 0
**************************************************************
```

Serializable.txt

```
**************************************************************
TypeA User Count : 10 TypeB User Count : 10
TypeA Deadlock Count : 686
TypeB Deadlock Count : 24
TypeA Average Time Cost : 00:05:09.8762694
TypeB Average Time Cost : 00:01:23.4352911
Other Exception Count : 0
**************************************************************

**************************************************************
TypeA User Count : 15 TypeB User Count : 15
TypeA Deadlock Count : 1126
TypeB Deadlock Count : 62
TypeA Average Time Cost : 00:07:38.2550071
TypeB Average Time Cost : 00:02:39.0244057
Other Exception Count : 0
**************************************************************

**************************************************************
TypeA User Count : 25 TypeB User Count : 25
TypeA Deadlock Count : 2037
TypeB Deadlock Count : 312
TypeA Average Time Cost : 00:14:23.9084298
TypeB Average Time Cost : 00:10:32.8554047
Other Exception Count : 0
**************************************************************

**************************************************************
TypeA User Count : 50 TypeB User Count : 50
TypeA Deadlock Count : 2824
TypeB Deadlock Count : 0
TypeA Average Time Cost : 00:27:22.2037996
TypeB Average Time Cost : 00:09:23.3688150
Other Exception Count : 0
**************************************************************

**************************************************************
TypeA User Count : 100 TypeB User Count : 100
TypeA Deadlock Count : 794
TypeB Deadlock Count : 0
TypeA Average Time Cost : 00:08:13.4602918
TypeB Average Time Cost : 00:02:54.5465862
Other Exception Count : 0
**************************************************************
```

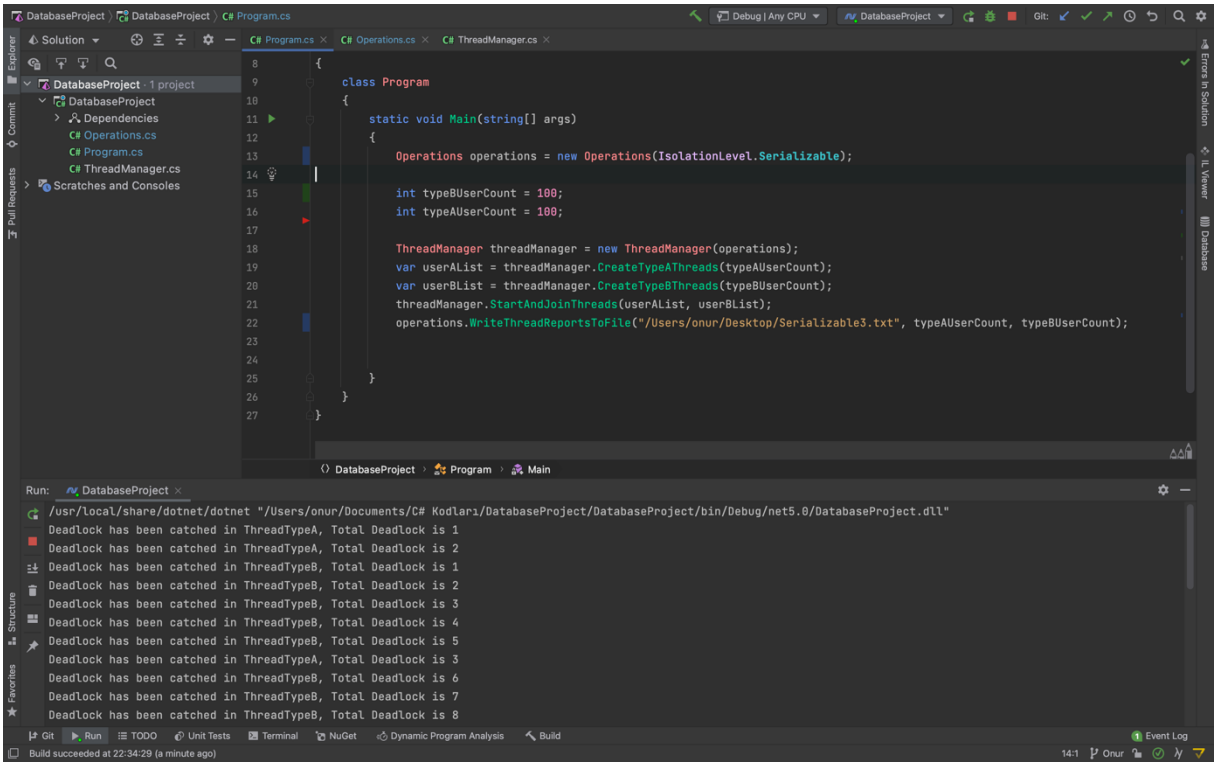
Serializable 50 and 100.txt

```
**************************************************************
TypeA User Count : 50 TypeB User Count : 50
TypeA Deadlock Count : 889
TypeB Deadlock Count : 261
TypeA Average Time Cost : 00:43:50.3854770
TypeB Average Time Cost : 00:13:35.9273109
Other Exception Count : 0
**************************************************************

**************************************************************
TypeA User Count : 100 TypeB User Count : 100
TypeA Deadlock Count : 98
TypeB Deadlock Count : 0
TypeA Average Time Cost : 00:50:47.9945593
TypeB Average Time Cost : 00:13:04.6241743
Other Exception Count : 0
**************************************************************

**************************************************************
TypeA User Count : 50 TypeB User Count : 50
TypeA Deadlock Count : 2634
TypeB Deadlock Count : 0
TypeA Average Time Cost : 00:27:28.2418881
TypeB Average Time Cost : 00:08:54.1124543
Other Exception Count : 0
**************************************************************

**************************************************************
TypeA User Count : 100 TypeB User Count : 100
TypeA Deadlock Count : 46
TypeB Deadlock Count : 0
TypeA Average Time Cost : 00:48:36.6062299
TypeB Average Time Cost : 00:10:10.0101698
Other Exception Count : 0
**************************************************************

**************************************************************
TypeA User Count : 50 TypeB User Count : 50
TypeA Deadlock Count : 1427
TypeB Deadlock Count : 112
TypeA Average Time Cost : 00:28:46.6131562
TypeB Average Time Cost : 00:01:24.2492233
Other Exception Count : 0
**************************************************************

**************************************************************
TypeA User Count : 100 TypeB User Count : 100
TypeA Deadlock Count : 57
TypeB Deadlock Count : 0
TypeA Average Time Cost : 00:04:48.6290962
TypeB Average Time Cost : 00:02:28.7544482
Other Exception Count : 0
**************************************************************
```

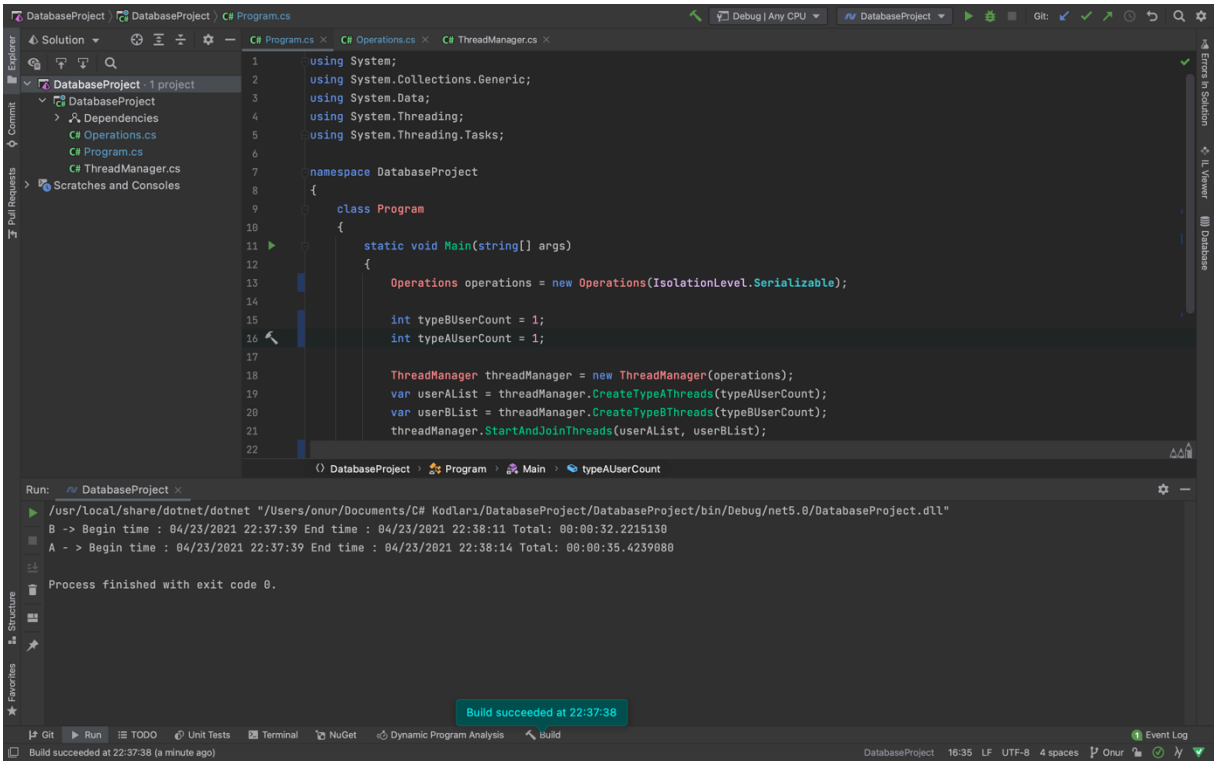
These outputs show the results of the extra tests we performed at the serializable level. These are the screenshots of the txt files created after our program runs.

Screenshots

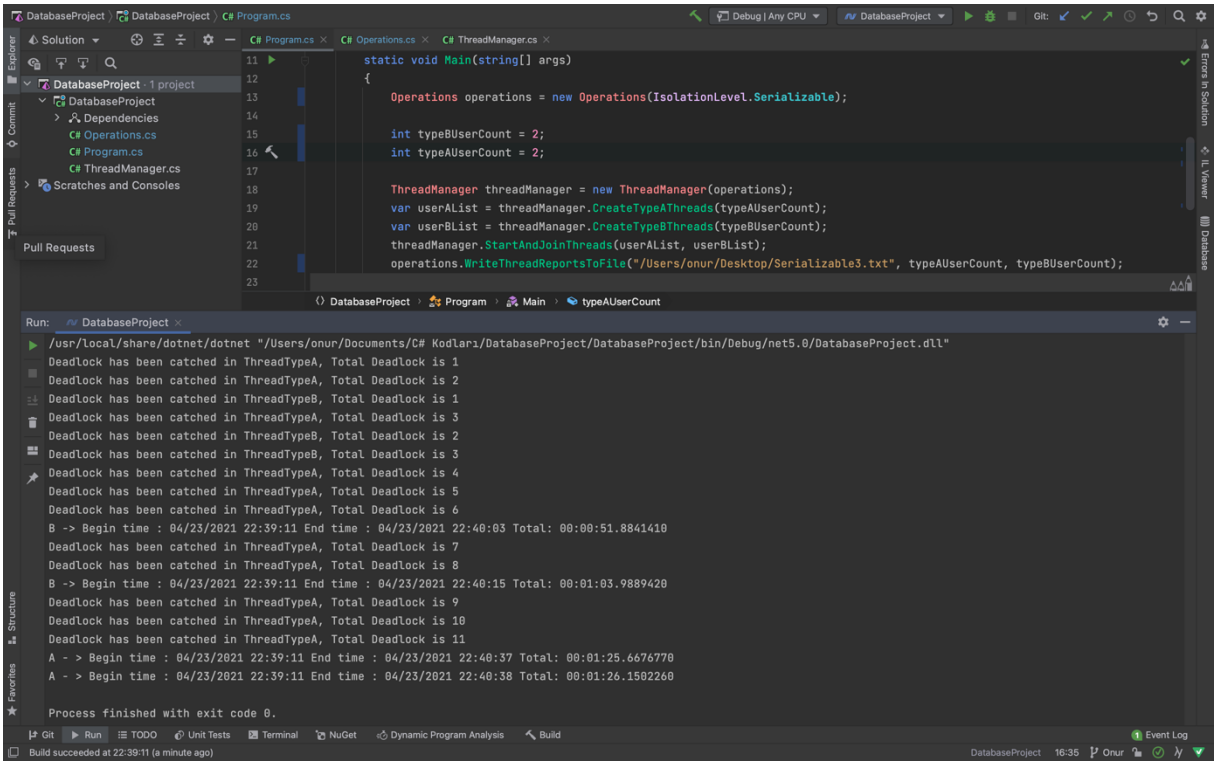
(Onur)



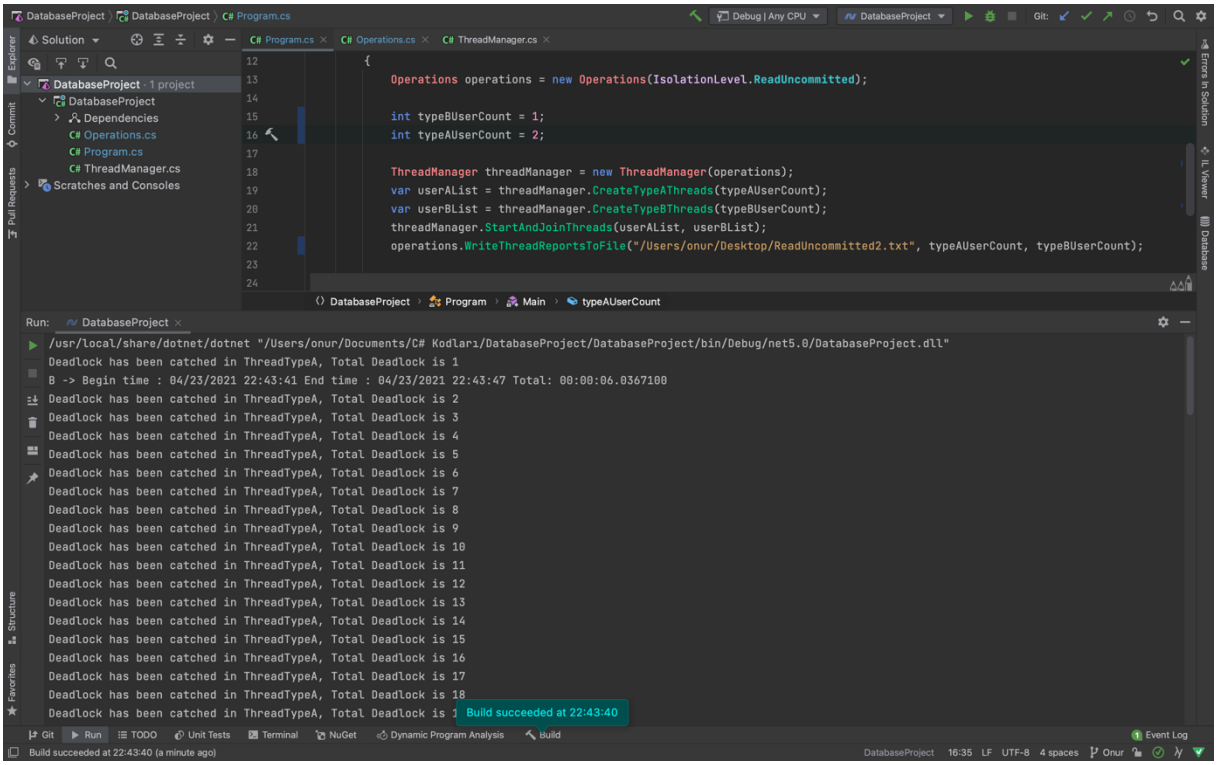
(Onur)



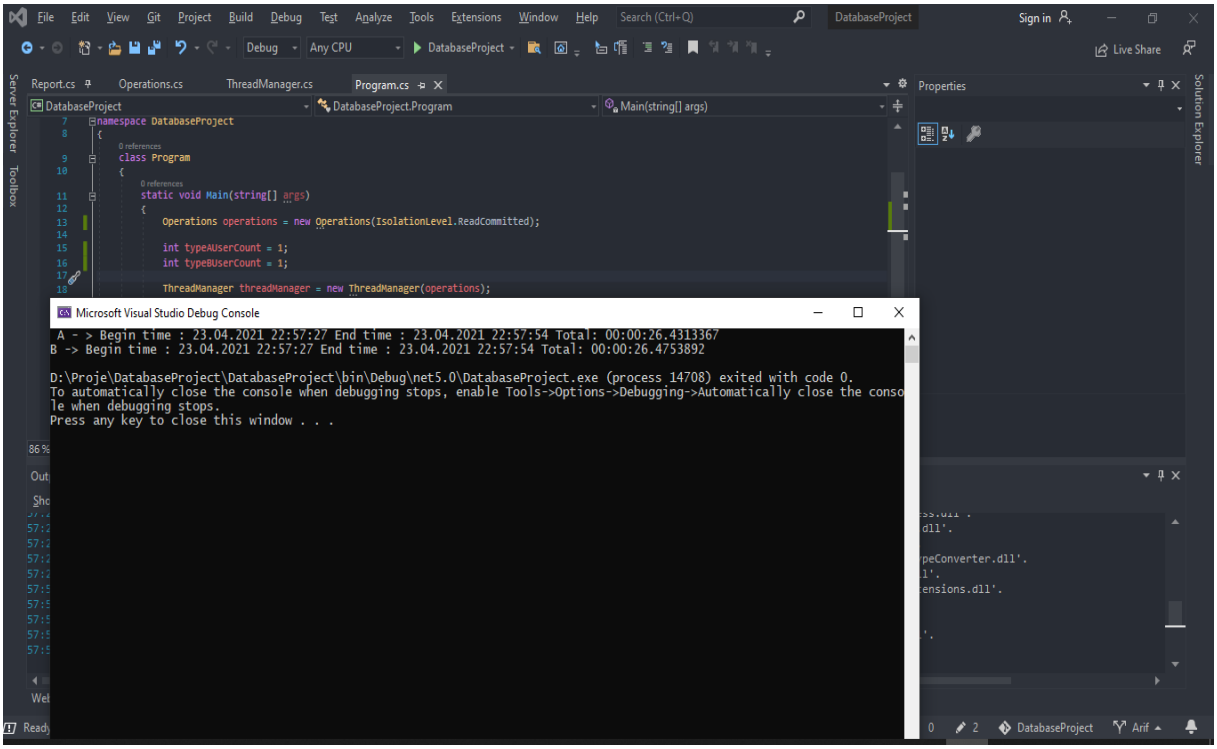
(Onur)


DatabaseProject > DatabaseProject > C# Program.cs
Debug | Any CPU
DatabaseProject
Git:
Solution
DatabaseProject · 1 project
DatabaseProject
Dependencies
Operations.cs
Program.cs
ThreadManager.cs
Scratches and Consoles
Pull Requests
C# Program.cs
C# Operations.cs
C# ThreadManager.cs
static void Main(string[] args)
{
Operations operations = new Operations(IsolationLevel.Serializable);
int typeBUserCount = 2;
int typeAUserCount = 2;
ThreadManager threadManager = new ThreadManager(operations);
var userAList = threadManager.CreateTypeAThreads(typeAUserCount);
var userBList = threadManager.CreateTypeBThreads(typeBUserCount);
threadManager.StartAndJoinThreads(userAList, userBList);
operations.WriteThreadReportsToFile("/Users/onur/Desktop/Serializable3.txt", typeAUserCount, typeBUserCount);
DatabaseProject > Program > Main > typeAUserCount
Run: DatabaseProject
/usr/local/share/dotnet/dotnet "/Users/onur/Documents/C# Kodları/DatabaseProject/DatabaseProject/bin/Debug/net5.0/DatabaseProject.dll"
Deadlock has been catched in ThreadTypeA, Total Deadlock is 1
Deadlock has been catched in ThreadTypeA, Total Deadlock is 2
Deadlock has been catched in ThreadTypeB, Total Deadlock is 1
Deadlock has been catched in ThreadTypeA, Total Deadlock is 3
Deadlock has been catched in ThreadTypeB, Total Deadlock is 2
Deadlock has been catched in ThreadTypeB, Total Deadlock is 3
Deadlock has been catched in ThreadTypeA, Total Deadlock is 4
Deadlock has been catched in ThreadTypeA, Total Deadlock is 5
Deadlock has been catched in ThreadTypeA, Total Deadlock is 6
B -> Begin time : 04/23/2021 22:39:11 End time : 04/23/2021 22:40:03 Total: 00:00:51.8841410
Deadlock has been catched in ThreadTypeA, Total Deadlock is 7
Deadlock has been catched in ThreadTypeA, Total Deadlock is 8
B -> Begin time : 04/23/2021 22:39:11 End time : 04/23/2021 22:40:15 Total: 00:01:03.9889420
Deadlock has been catched in ThreadTypeA, Total Deadlock is 9
Deadlock has been catched in ThreadTypeA, Total Deadlock is 10
Deadlock has been catched in ThreadTypeA, Total Deadlock is 11
A - > Begin time : 04/23/2021 22:39:11 End time : 04/23/2021 22:40:37 Total: 00:01:25.6676770
A - > Begin time : 04/23/2021 22:39:11 End time : 04/23/2021 22:40:38 Total: 00:01:26.1502260
Process finished with exit code 0.
Build succeeded at 22:39:11 (a minute ago)

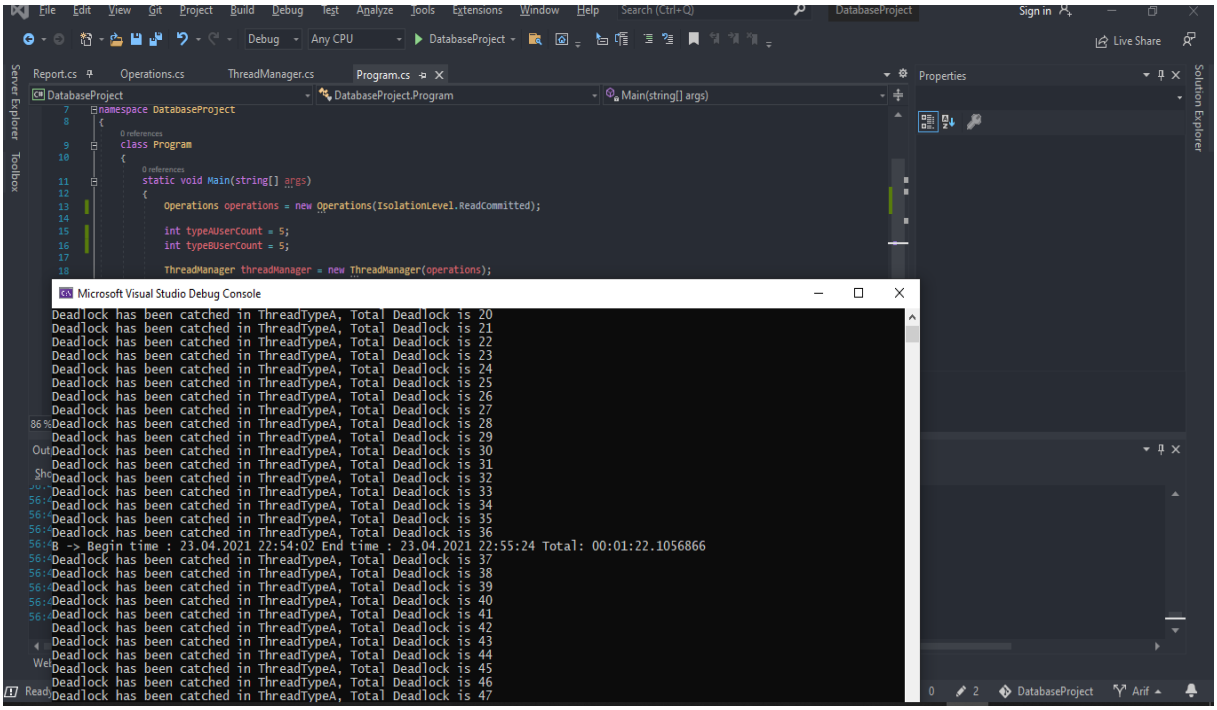
(Onur)


DatabaseProject > DatabaseProject > C# Program.cs
Debug | Any CPU
DatabaseProject
Git:
Solution
DatabaseProject · 1 project
DatabaseProject
Dependencies
Operations.cs
Program.cs
ThreadManager.cs
Scratches and Consoles
C# Program.cs
C# Operations.cs
C# ThreadManager.cs
{
Operations operations = new Operations(IsolationLevel.ReadUncommitted);
int typeBUserCount = 1;
int typeAUserCount = 2;
ThreadManager threadManager = new ThreadManager(operations);
var userAList = threadManager.CreateTypeAThreads(typeAUserCount);
var userBList = threadManager.CreateTypeBThreads(typeBUserCount);
threadManager.StartAndJoinThreads(userAList, userBList);
operations.WriteThreadReportsToFile("/Users/onur/Desktop/ReadUncommitted2.txt", typeAUserCount, typeBUserCount);
DatabaseProject > Program > Main > typeAUserCount
Run: DatabaseProject
/usr/local/share/dotnet/dotnet "/Users/onur/Documents/C# Kodları/DatabaseProject/DatabaseProject/bin/Debug/net5.0/DatabaseProject.dll"
Deadlock has been catched in ThreadTypeA, Total Deadlock is 1
B -> Begin time : 04/23/2021 22:43:41 End time : 04/23/2021 22:43:47 Total: 00:00:06.0367100
Deadlock has been catched in ThreadTypeA, Total Deadlock is 2
Deadlock has been catched in ThreadTypeA, Total Deadlock is 3
Deadlock has been catched in ThreadTypeA, Total Deadlock is 4
Deadlock has been catched in ThreadTypeA, Total Deadlock is 5
Deadlock has been catched in ThreadTypeA, Total Deadlock is 6
Deadlock has been catched in ThreadTypeA, Total Deadlock is 7
Deadlock has been catched in ThreadTypeA, Total Deadlock is 8
Deadlock has been catched in ThreadTypeA, Total Deadlock is 9
Deadlock has been catched in ThreadTypeA, Total Deadlock is 10
Deadlock has been catched in ThreadTypeA, Total Deadlock is 11
Deadlock has been catched in ThreadTypeA, Total Deadlock is 12
Deadlock has been catched in ThreadTypeA, Total Deadlock is 13
Deadlock has been catched in ThreadTypeA, Total Deadlock is 14
Deadlock has been catched in ThreadTypeA, Total Deadlock is 15
Deadlock has been catched in ThreadTypeA, Total Deadlock is 16
Deadlock has been catched in ThreadTypeA, Total Deadlock is 17
Deadlock has been catched in ThreadTypeA, Total Deadlock is 18
Deadlock has been catched in ThreadTypeA, Total Deadlock is 1
Build succeeded at 22:43:40
Build succeeded at 22:43:40 (a minute ago)

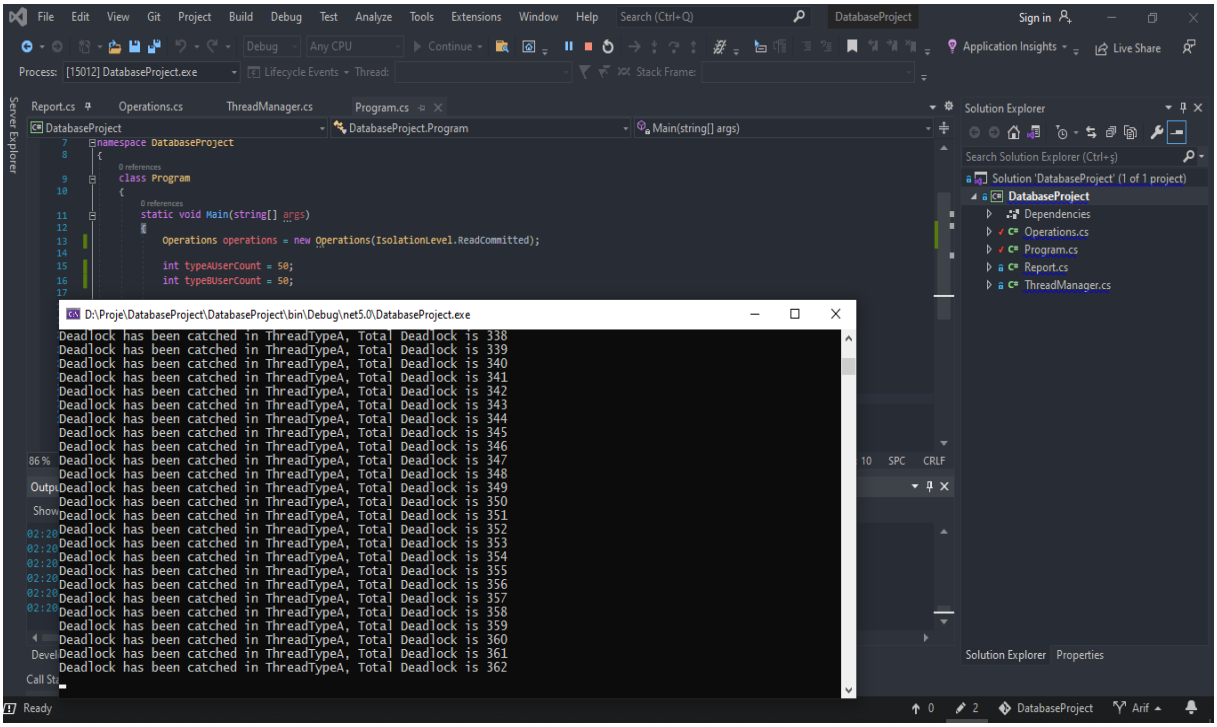
(Arif)


Report.cs
Operations.cs
ThreadManager.cs
Program.cs
DatabaseProject
DatabaseProject.Program
Main(string[] args)
namespace DatabaseProject
{
class Program
{
static void Main(string[] args)
{
Operations operations = new Operations(IsolationLevel.ReadCommitted);
int typeAUserCount = 1;
int typeBUserCount = 1;
ThreadManager threadManager = new ThreadManager(operations);
Microsoft Visual Studio Debug Console
A - > Begin time : 23.04.2021 22:57:27 End time : 23.04.2021 22:57:54 Total: 00:00:26.4313367
B -> Begin time : 23.04.2021 22:57:27 End time : 23.04.2021 22:57:54 Total: 00:00:26.4753892
D:\Proje\DatabaseProject\DatabaseProject\bin\Debug\net5.0\DatabaseProject.exe (process 14708) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .

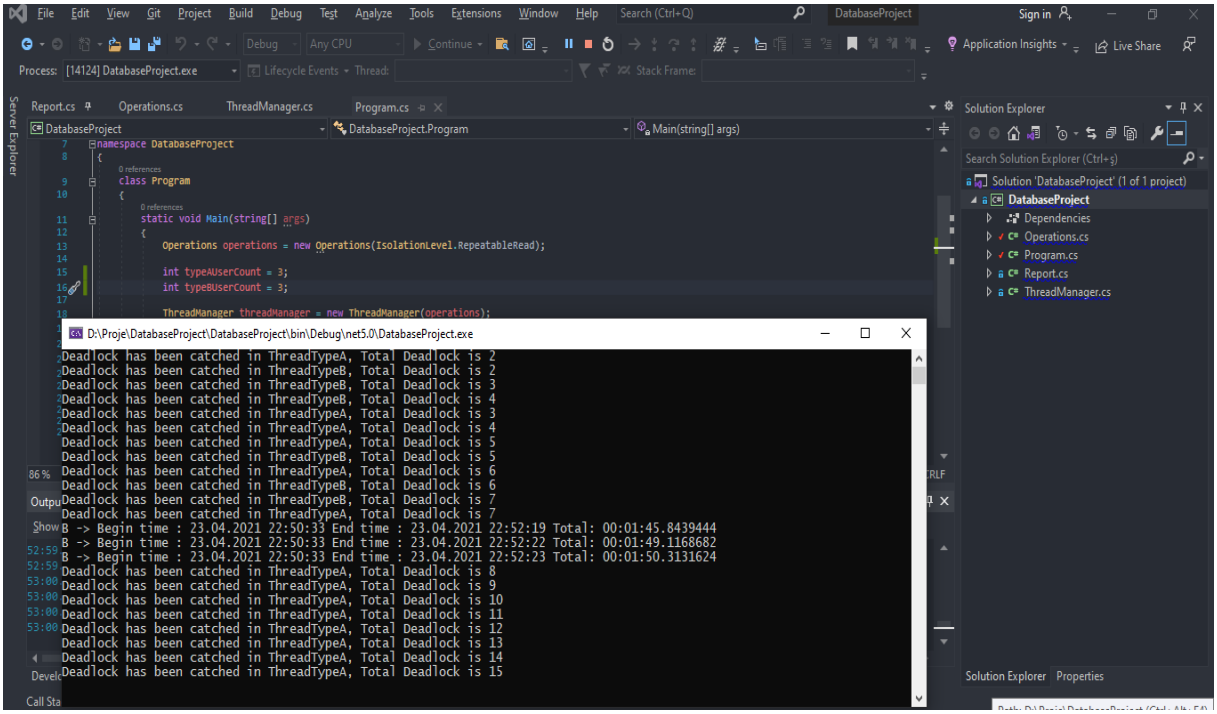
(Arif)


Report.cs
Operations.cs
ThreadManager.cs
Program.cs
DatabaseProject
DatabaseProject.Program
Main(string[] args)
namespace DatabaseProject
{
class Program
{
static void Main(string[] args)
{
Operations operations = new Operations(IsolationLevel.ReadCommitted);
int typeAUserCount = 5;
int typeBUserCount = 5;
ThreadManager threadManager = new ThreadManager(operations);
Microsoft Visual Studio Debug Console
Deadlock has been catched in ThreadTypeA, Total Deadlock is 20
Deadlock has been catched in ThreadTypeA, Total Deadlock is 21
Deadlock has been catched in ThreadTypeA, Total Deadlock is 22
Deadlock has been catched in ThreadTypeA, Total Deadlock is 23
Deadlock has been catched in ThreadTypeA, Total Deadlock is 24
Deadlock has been catched in ThreadTypeA, Total Deadlock is 25
Deadlock has been catched in ThreadTypeA, Total Deadlock is 26
Deadlock has been catched in ThreadTypeA, Total Deadlock is 27
Deadlock has been catched in ThreadTypeA, Total Deadlock is 28
Deadlock has been catched in ThreadTypeA, Total Deadlock is 29
Deadlock has been catched in ThreadTypeA, Total Deadlock is 30
Deadlock has been catched in ThreadTypeA, Total Deadlock is 31
Deadlock has been catched in ThreadTypeA, Total Deadlock is 32
Deadlock has been catched in ThreadTypeA, Total Deadlock is 33
Deadlock has been catched in ThreadTypeA, Total Deadlock is 34
Deadlock has been catched in ThreadTypeA, Total Deadlock is 35
Deadlock has been catched in ThreadTypeA, Total Deadlock is 36
B -> Begin time : 23.04.2021 22:54:02 End time : 23.04.2021 22:55:24 Total: 00:01:22.1056866
Deadlock has been catched in ThreadTypeA, Total Deadlock is 37
Deadlock has been catched in ThreadTypeA, Total Deadlock is 38
Deadlock has been catched in ThreadTypeA, Total Deadlock is 39
Deadlock has been catched in ThreadTypeA, Total Deadlock is 40
Deadlock has been catched in ThreadTypeA, Total Deadlock is 41
Deadlock has been catched in ThreadTypeA, Total Deadlock is 42
Deadlock has been catched in ThreadTypeA, Total Deadlock is 43
Deadlock has been catched in ThreadTypeA, Total Deadlock is 44
Deadlock has been catched in ThreadTypeA, Total Deadlock is 45
Deadlock has been catched in ThreadTypeA, Total Deadlock is 46
Deadlock has been catched in ThreadTypeA, Total Deadlock is 47

(Arif)


Process: [15012] DatabaseProject.exe
Report.cs
Operations.cs
ThreadManager.cs
Program.cs
DatabaseProject
DatabaseProject.Program
Main(string[] args)
namespace DatabaseProject
{
class Program
{
static void Main(string[] args)
{
Operations operations = new Operations(IsolationLevel.ReadCommitted);
int typeAUserCount = 50;
int typeBUserCount = 50;
D:\Proje\DatabaseProject\DatabaseProject\bin\Debug\net5.0\DatabaseProject.exe
Deadlock has been catched in ThreadTypeA, Total Deadlock is 338
Deadlock has been catched in ThreadTypeA, Total Deadlock is 339
Deadlock has been catched in ThreadTypeA, Total Deadlock is 340
Deadlock has been catched in ThreadTypeA, Total Deadlock is 341
Deadlock has been catched in ThreadTypeA, Total Deadlock is 342
Deadlock has been catched in ThreadTypeA, Total Deadlock is 343
Deadlock has been catched in ThreadTypeA, Total Deadlock is 344
Deadlock has been catched in ThreadTypeA, Total Deadlock is 345
Deadlock has been catched in ThreadTypeA, Total Deadlock is 346
Deadlock has been catched in ThreadTypeA, Total Deadlock is 347
Deadlock has been catched in ThreadTypeA, Total Deadlock is 348
Deadlock has been catched in ThreadTypeA, Total Deadlock is 349
Deadlock has been catched in ThreadTypeA, Total Deadlock is 350
Deadlock has been catched in ThreadTypeA, Total Deadlock is 351
Deadlock has been catched in ThreadTypeA, Total Deadlock is 352
Deadlock has been catched in ThreadTypeA, Total Deadlock is 353
Deadlock has been catched in ThreadTypeA, Total Deadlock is 354
Deadlock has been catched in ThreadTypeA, Total Deadlock is 355
Deadlock has been catched in ThreadTypeA, Total Deadlock is 356
Deadlock has been catched in ThreadTypeA, Total Deadlock is 357
Deadlock has been catched in ThreadTypeA, Total Deadlock is 358
Deadlock has been catched in ThreadTypeA, Total Deadlock is 359
Deadlock has been catched in ThreadTypeA, Total Deadlock is 360
Deadlock has been catched in ThreadTypeA, Total Deadlock is 361
Deadlock has been catched in ThreadTypeA, Total Deadlock is 362

(Arif)


Process: [14124] DatabaseProject.exe
Report.cs
Operations.cs
ThreadManager.cs
Program.cs
DatabaseProject
DatabaseProject.Program
Main(string[] args)
namespace DatabaseProject
{
class Program
{
static void Main(string[] args)
{
Operations operations = new Operations(IsolationLevel.RepeatableRead);
int typeAUserCount = 3;
int typeBUserCount = 3;
ThreadManager threadManager = new ThreadManager(operations);
D:\Proje\DatabaseProject\DatabaseProject\bin\Debug\net5.0\DatabaseProject.exe
Deadlock has been catched in ThreadTypeA, Total Deadlock is 2
Deadlock has been catched in ThreadTypeB, Total Deadlock is 2
Deadlock has been catched in ThreadTypeB, Total Deadlock is 3
Deadlock has been catched in ThreadTypeB, Total Deadlock is 4
Deadlock has been catched in ThreadTypeA, Total Deadlock is 3
Deadlock has been catched in ThreadTypeA, Total Deadlock is 4
Deadlock has been catched in ThreadTypeA, Total Deadlock is 5
Deadlock has been catched in ThreadTypeB, Total Deadlock is 5
Deadlock has been catched in ThreadTypeA, Total Deadlock is 6
Deadlock has been catched in ThreadTypeB, Total Deadlock is 6
Deadlock has been catched in ThreadTypeB, Total Deadlock is 7
Deadlock has been catched in ThreadTypeA, Total Deadlock is 7
B -> Begin time : 23.04.2021 22:50:33 End time : 23.04.2021 22:52:19 Total: 00:01:45.8439444
B -> Begin time : 23.04.2021 22:50:33 End time : 23.04.2021 22:52:22 Total: 00:01:49.1168682
B -> Begin time : 23.04.2021 22:50:33 End time : 23.04.2021 22:52:23 Total: 00:01:50.3131624
Deadlock has been catched in ThreadTypeA, Total Deadlock is 8
Deadlock has been catched in ThreadTypeA, Total Deadlock is 9
Deadlock has been catched in ThreadTypeA, Total Deadlock is 10
Deadlock has been catched in ThreadTypeA, Total Deadlock is 11
Deadlock has been catched in ThreadTypeA, Total Deadlock is 12
Deadlock has been catched in ThreadTypeA, Total Deadlock is 13
Deadlock has been catched in ThreadTypeA, Total Deadlock is 14
Deadlock has been catched in ThreadTypeA, Total Deadlock is 15

Environments

Onur Akalın

MacOS – SQL2019 as a Docker Container – Rider IDE Used

Processor: Intel® Core™ i5-8257U 1.40 Ghz, up to 3.90 GHz

Ram: 8 Gb

Ahmet Arif Özçelik

Windows – SQL 2019 – Visual Studio IDE Used

Processor: Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz

Ram: 16 Gb

Source Code

Program.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Threading;
using System.Threading.Tasks;

namespace DatabaseProject
{
    class Program
    {
        static void Main(string[] args)
        {
            Operations operations = new Operations(IsolationLevel.ReadUncommitted);

            int typeBUserCount = 1;
            int typeAUserCount = 2;

            ThreadManager threadManager = new ThreadManager(operations);
            var userAList = threadManager.CreateTypeAThreads(typeAUserCount);
            var userBList = threadManager.CreateTypeBThreads(typeBUserCount);
            threadManager.StartAndJoinThreads(userAList, userBList);
            operations.WriteThreadReportsToFile("/Users/onur/Desktop/ReadUncommitted2.txt", typeAUserCount, typeBUserCount);



        }
    }
}
```

Operations.cs

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.IO;

namespace DatabaseProject
{
   public class Operations
   {
      private readonly IsolationLevel _isolationLevel;
      private int _typeADeadlockCount;
      private int _typeBDeadlockCount;
      private int _otherExceptionsCount;
      private TimeSpan _typeATotalTime;
      private TimeSpan _typeBTotalTime;
      private readonly object _threadLock;

      private string _connectionString =
         @"Data Source=localhost;Initial Catalog=AdventureWorks2012;User ID=sa;Password=Onur-1234;Connection Timeout = 6000";

      public Operations(IsolationLevel isolationLevel)
      {
         _isolationLevel = isolationLevel;
         _typeADeadlockCount = 0;
         _typeBDeadlockCount = 0;
         _otherExceptionsCount = 0;
         _typeATotalTime = TimeSpan.Zero;
         _typeBTotalTime = TimeSpan.Zero;
         _threadLock = new object();
      }

      private SqlCommand UpdateQuery(string beginDate, string endDate, SqlConnection connection,
         SqlTransaction transaction)
      {
         SqlCommand command = new SqlCommand(
            "UPDATE Sales.SalesOrderDetail " +
            "SET UnitPrice = UnitPrice * 10.0 / 10.0 " +
            "WHERE UnitPrice > 100 " +
            "AND EXISTS (SELECT * FROM Sales.SalesOrderHeader " +
            "WHERE Sales.SalesOrderHeader.SalesOrderID = " +
            "Sales.SalesOrderDetail.SalesOrderID " +
            "AND Sales.SalesOrderHeader.OrderDate " +
            "BETWEEN @BeginDate AND @EndDate " +
            "AND Sales.SalesOrderHeader.OnlineOrderFlag = 1)",
            connection, transaction);

         command.Parameters.AddWithValue("@BeginDate", beginDate);
         command.Parameters.AddWithValue("@EndDate", endDate);

         return command;
      }

      private SqlCommand SelectQuery(string beginDate, string endDate, SqlConnection connection,
         SqlTransaction transaction)
      {
         SqlCommand command = new SqlCommand(
            "SELECT SUM(Sales.SalesOrderDetail.OrderQty) " +
            "FROM Sales.SalesOrderDetail " +
            "WHERE UnitPrice > 100 " +
            "AND EXISTS (SELECT * FROM Sales.SalesOrderHeader " +
            "WHERE Sales.SalesOrderHeader.SalesOrderID = " +
            "Sales.SalesOrderDetail.SalesOrderID " +
            "AND Sales.SalesOrderHeader.OrderDate " +
            "BETWEEN @BeginDate AND @EndDate " +
            "AND Sales.SalesOrderHeader.OnlineOrderFlag = 1)",
            connection, transaction);

         command.Parameters.AddWithValue("@BeginDate", beginDate);
         command.Parameters.AddWithValue("@EndDate", endDate);
```

```
    return command;
}

public void ThreadTypeA()
{
    SqlConnection conn = new SqlConnection(_connectionString);
    SqlTransaction transaction = null;
    DateTime beginTime = DateTime.Now;
    for (int i = 0; i < 100; i++)
    {
        try
        {
            conn.Open();
            transaction = conn.BeginTransaction(_isolationLevel);

            Random random = new Random();

            if (random.NextDouble() < 0.5)
                UpdateQuery("20110101", "20111231", conn, transaction).ExecuteNonQuery();
            if (random.NextDouble() < 0.5)
                UpdateQuery("20120101", "20121231", conn, transaction).ExecuteNonQuery();
            if (random.NextDouble() < 0.5)
                UpdateQuery("20130101", "20131231", conn, transaction).ExecuteNonQuery();
            if (random.NextDouble() < 0.5)
                UpdateQuery("20140101", "20141231", conn, transaction).ExecuteNonQuery();
            if (random.NextDouble() < 0.5)
                UpdateQuery("20150101", "20151231", conn, transaction).ExecuteNonQuery();

            transaction.Commit();
            conn.Close();
        }
        catch (SqlException ex1)
        {
            if (ex1.Number == 1205)
            {
                try
                {
                    lock (_threadLock)
                    {
                        _typeADeadlockCount++;
                    }

                    Console.WriteLine("Deadlock has been catched in ThreadTypeA, Total Deadlock is {0}", _typeADeadlockCount);
                    transaction.Rollback();
                }
                catch (Exception ex2)
                {
                    Console.WriteLine("ThreadTypeA ,Rollback Exception Type: {0}", ex2.GetType());
                    lock (_threadLock)
                    {
                        _otherExceptionsCount++;
                    }
                    transaction.Rollback();
                }
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            lock (_threadLock)
            {
                _otherExceptionsCount++;
            }
            transaction.Rollback();
        }
```

```
        finally
        {
            if (conn.State == ConnectionState.Open)
            {
                conn.Close();
            }
        }
    }

    DateTime endTime = DateTime.Now;
    TimeSpan elapsed = endTime - beginTime;
    Console.WriteLine("A - > Begin time : " +  beginTime + " End time : " + endTime  + " Total: " + elapsed);
    lock (_threadLock)
    {
        _typeATotalTime += elapsed;
    }
}

public void ThreadTypeB()
{
    SqlConnection conn = new SqlConnection(_connectionString);
    SqlTransaction transaction = null;
    DateTime beginTime = DateTime.Now;

    for (int i = 0; i < 100; i++)
    {
        try
        {
            conn.Open();
            transaction = conn.BeginTransaction(_isolationLevel);

            Random random = new Random();

            if (random.NextDouble() < 0.5)
                SelectQuery("20110101", "20111231", conn, transaction).ExecuteNonQuery();
            if (random.NextDouble() < 0.5)
                SelectQuery("20120101", "20121231", conn, transaction).ExecuteNonQuery();
            if (random.NextDouble() < 0.5)
                SelectQuery("20130101", "20131231", conn, transaction).ExecuteNonQuery();
            if (random.NextDouble() < 0.5)
                SelectQuery("20140101", "20141231", conn, transaction).ExecuteNonQuery();
            if (random.NextDouble() < 0.5)
                SelectQuery("20150101", "20151231", conn, transaction).ExecuteNonQuery();

            transaction.Commit();
            conn.Close();
        }
        catch (SqlException ex1)
        {
            if (ex1.Number == 1205)
            {
                try
                {
                    lock (_threadLock)
                    {
                        _typeBDeadlockCount++;
                    }

                    Console.WriteLine("Deadlock has been catched in ThreadTypeB, Total Deadlock is {0}", _typeBDeadlockCount);
                    transaction.Rollback();
                }
```

```
                catch (Exception ex2)
                {
                    Console.WriteLine("ThreadTypeB ,Rollback Exception Type: {0}", ex2.GetType());
                    Console.WriteLine("Message: {0}", ex2.Message);
                    lock (_threadLock)
                    {
                        _otherExceptionsCount++;
                    }
                    transaction.Rollback();
                }
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            lock (_threadLock)
            {
                _otherExceptionsCount++;
            }
            transaction.Rollback();
        }
        finally
        {
            if (conn.State == ConnectionState.Open)
            {
                conn.Close();
            }
        }
    }

    DateTime endTime = DateTime.Now;
    TimeSpan elapsed = endTime - beginTime;
    Console.WriteLine("B -> Begin time : " +  beginTime + " End time : " + endTime  + " Total: " + elapsed);
    lock (_threadLock)
    {
        _typeBTotalTime += elapsed;
    }
}

public void WriteThreadReportsToFile(string fileName, int typeAUserCount, int typeBUserCount)
{
    string[] lines =
    {
        "*****************************************************************",
        "TypeA User Count : " + typeAUserCount + " TypeB User Count : " + typeBUserCount,
        "TypeA Deadlock Count : " + _typeADeadlockCount,
        "TypeB Deadlock Count : " + _typeBDeadlockCount,
        "TypeA Average Time Cost : " + (_typeATotalTime / typeAUserCount),
        "TypeB Average Time Cost : " + (_typeBTotalTime / typeBUserCount),
        "Other Exception Count : " + _otherExceptionsCount,
        "*****************************************************************\n"
    };
    File.AppendAllLines(fileName, lines);
}
}
}
```

ThreadManager.cs

```
using System.Collections.Generic;
using System.Threading;

namespace DatabaseProject
{
    public class ThreadManager
    {
        private readonly Operations _operations;
```

```
    public ThreadManager(Operations operations)
    {
        _operations = operations;
    }

    public List<Thread> CreateTypeAThreads(int threadNumber)
    {
        List<Thread> threadListA = new List<Thread>();
        for (int i = 0; i < threadNumber; i++)
        {
            threadListA.Add(new Thread(new ThreadStart(_operations.ThreadTypeA)));
        }
        return threadListA;
    }

    public List<Thread> CreateTypeBThreads(int threadNumber)
    {
        List<Thread> threadListB = new List<Thread>();
        for (int i = 0; i < threadNumber; i++)
        {
            threadListB.Add(new Thread(new ThreadStart(_operations.ThreadTypeB)));
        }
        return threadListB;
    }

    public void StartAndJoinThreads(List<Thread> threadListTypeA, List<Thread> threadListTypeB)
    {
        foreach (var thread in threadListTypeA)
        {
            thread.Start();
        }

        foreach (var thread in threadListTypeB)
        {
            thread.Start();
        }

        foreach (var thread in threadListTypeA)
        {
            thread.Join();
        }

        foreach (var thread in threadListTypeB)
        {
            thread.Join();
        }
    }
  }
}
```

Onur Akalın 170706018 & Ahmet Arif Özçelik 170706016