Report 2

Databases optimization

Attention! Whole project is available at the repository.

Introduction

The whole process of testing rely on using a special class that is responsible for execution concrete attempt. That class is TestAttemptHandler. Each user has its own database connection representing by MicrosoftSQLDatabase that implements IDatabase interface. Every user derives from an abstract class called AbstrctUser.

1. Concurrency – solution 1

a. Users

i. BrewerFirst
ii. BrewerSecond

b. Test method

i. `OptimicticConcurrency()`

c. Output

[Click link below]
Figure 1: Output for Concurrency – solution 1

d. Description

Both breweries wanted to update the alcohol percentage value of the same beer. They have done this at the same time and work of the second one was just overwritten.

e. Solution

First solution of that concurrency problem is to use optimistic way. This mean that program should save values of the record on the beginning of update operation. When user want to confirm his operation, script should compare values stored in his memory with currently downloaded from database. If data are different, program will pop up window with warning that someone just edited this row.

2. Access points amount at the same time

a. Users

i. ConnectionTester

b. Test method

i. `ConnectionPointsAmount()`

c. Output:

[Click link below]
Figure 2: Output for Output for Access points amount at the same time

d. Description

When I tested what happens if I reach a maximum amount of open connections to the database I noticed that it work like a queue. Max connections number for my database is about 100. Then the database waits till some of the connections disconnect and allow next one from the queue to connect.

e. Solution

In my opinion, the solution provided by Microsoft is good enough. It doesn’t cause any errors about what is wanted behaviour. One thing that could be improved, the database should send the message that you are for example fifth at the queue to connect.

3. Concurrency (solution 2)

a. Users

i. BreweryOwner
ii. DatabaseAdmin

b. Test method

i. `PessimisticConcurrency()`

c. Output:

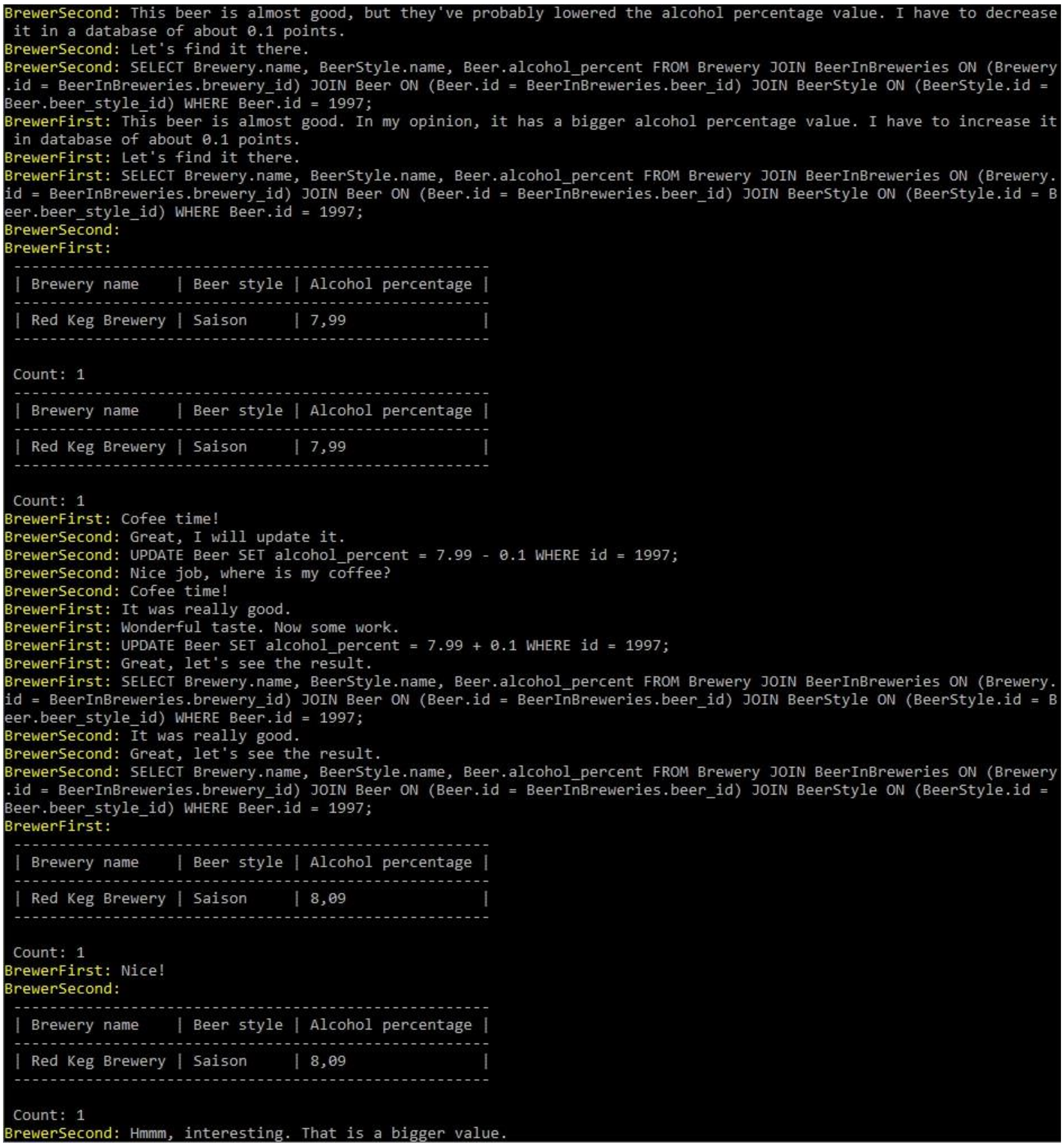
[Click link below]
Figure 3: Output for Concurrency (solution 2)

d. Description

These users want to edit the same record. They find one that they expect, but then Brewery owner change name of Brewery. DatabaseAdmin doesn’t know that so he deletes brewery that he found, but in fact, he did nothing, because that record doesn’t exist anymore because BreweryOwner edited it.

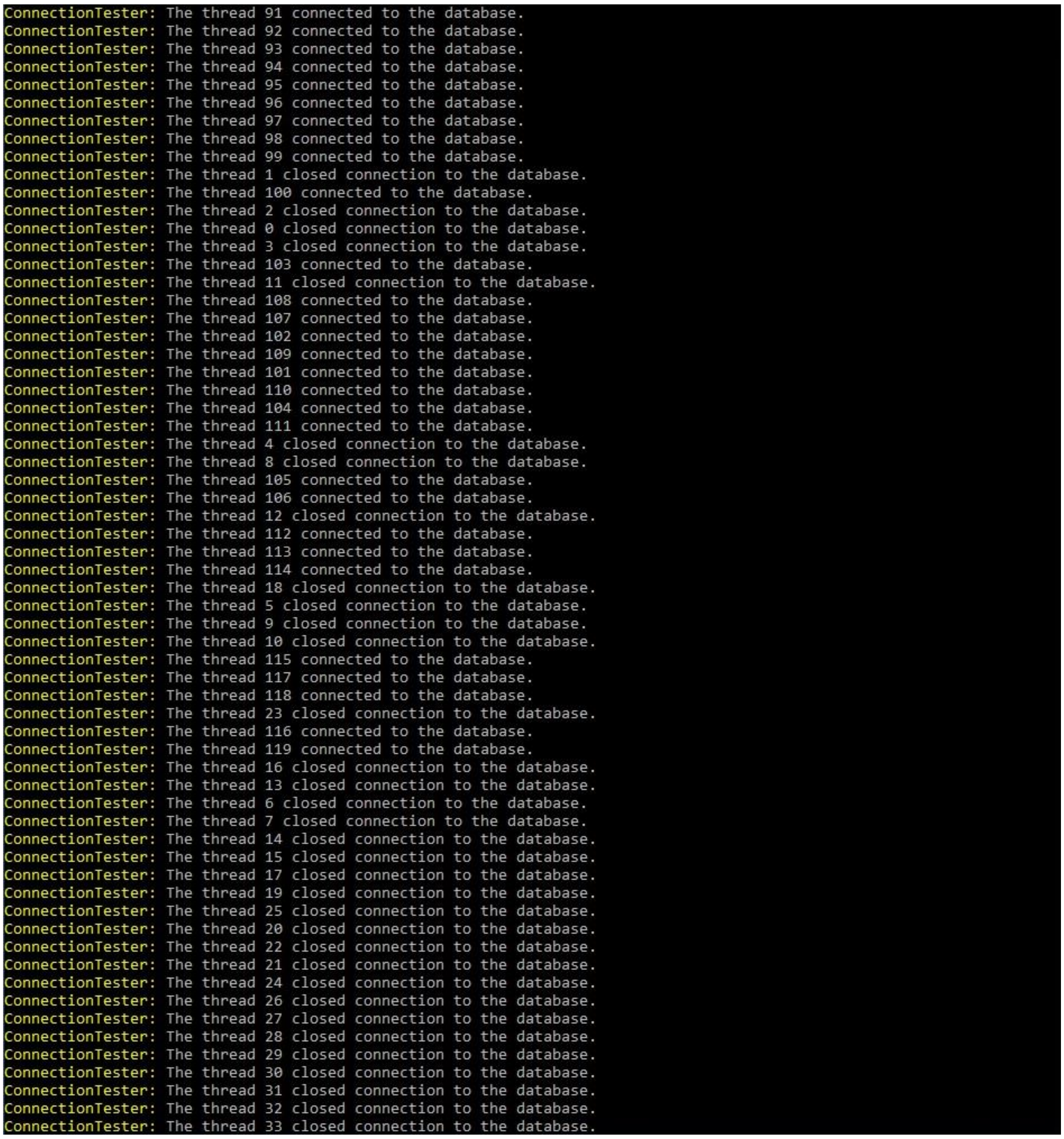
e. Solution

Another solution for concurrency is to implement pessimistic variant. If an application does need to prevent accidental data loss in concurrency scenarios, one way to do that is to use database locks. For example, before you read a row from a database, you request a lock for read-only or for update access. If you lock a row for update access, no other users are allowed to lock the row either for read-only or update access, because they would get a copy of data that's in the process of being changed. If you lock a row for read-only access, others can also lock it for read-only access but not for the update.



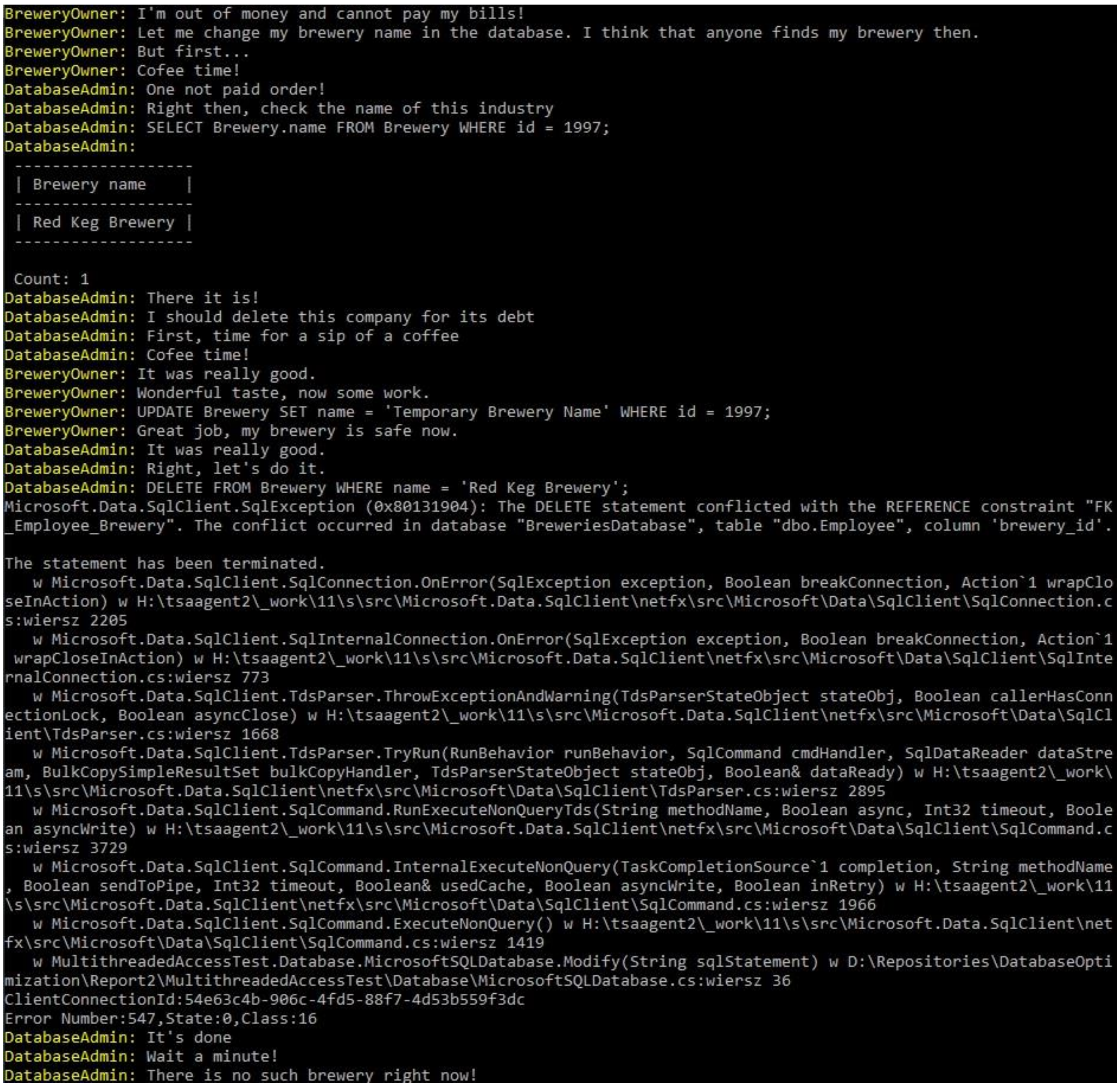
```
BrewerSecond: This beer is almost good, but they've probably lowered the alcohol percentage value. I have to decrease
 it in a database of about 0.1 points.
BrewerSecond: Let's find it there.
BrewerSecond: SELECT Brewery.name, BeerStyle.name, Beer.alcohol_percent FROM Brewery JOIN BeerInBreweries ON (Brewery
.id = BeerInBreweries.brewery_id) JOIN Beer ON (Beer.id = BeerInBreweries.beer_id) JOIN BeerStyle ON (BeerStyle.id =
Beer.beer_style_id) WHERE Beer.id = 1997;
BrewerFirst: This beer is almost good. In my opinion, it has a bigger alcohol percentage value. I have to increase it
 in database of about 0.1 points.
BrewerFirst: Let's find it there.
BrewerFirst: SELECT Brewery.name, BeerStyle.name, Beer.alcohol_percent FROM Brewery JOIN BeerInBreweries ON (Brewery.
id = BeerInBreweries.brewery_id) JOIN Beer ON (Beer.id = BeerInBreweries.beer_id) JOIN BeerStyle ON (BeerStyle.id = B
eer.beer_style_id) WHERE Beer.id = 1997;
BrewerSecond:
BrewerFirst:
 ----------------------------------------------------
 | Brewery name    | Beer style | Alcohol percentage |
 ----------------------------------------------------
 | Red Keg Brewery | Saison     | 7,99               |
 ----------------------------------------------------

 Count: 1
 ----------------------------------------------------
 | Brewery name    | Beer style | Alcohol percentage |
 ----------------------------------------------------
 | Red Keg Brewery | Saison     | 7,99               |
 ----------------------------------------------------

 Count: 1
BrewerFirst: Cofee time!
BrewerSecond: Great, I will update it.
BrewerSecond: UPDATE Beer SET alcohol_percent = 7.99 - 0.1 WHERE id = 1997;
BrewerSecond: Nice job, where is my coffee?
BrewerSecond: Cofee time!
BrewerFirst: It was really good.
BrewerFirst: Wonderful taste. Now some work.
BrewerFirst: UPDATE Beer SET alcohol_percent = 7.99 + 0.1 WHERE id = 1997;
BrewerFirst: Great, let's see the result.
BrewerFirst: SELECT Brewery.name, BeerStyle.name, Beer.alcohol_percent FROM Brewery JOIN BeerInBreweries ON (Brewery.
id = BeerInBreweries.brewery_id) JOIN Beer ON (Beer.id = BeerInBreweries.beer_id) JOIN BeerStyle ON (BeerStyle.id = B
eer.beer_style_id) WHERE Beer.id = 1997;
BrewerSecond: It was really good.
BrewerSecond: Great, let's see the result.
BrewerSecond: SELECT Brewery.name, BeerStyle.name, Beer.alcohol_percent FROM Brewery JOIN BeerInBreweries ON (Brewery
.id = BeerInBreweries.brewery_id) JOIN Beer ON (Beer.id = BeerInBreweries.beer_id) JOIN BeerStyle ON (BeerStyle.id =
Beer.beer_style_id) WHERE Beer.id = 1997;
BrewerFirst:
 ----------------------------------------------------
 | Brewery name    | Beer style | Alcohol percentage |
 ----------------------------------------------------
 | Red Keg Brewery | Saison     | 8,09               |
 ----------------------------------------------------

 Count: 1
BrewerFirst: Nice!
BrewerSecond:
 ----------------------------------------------------
 | Brewery name    | Beer style | Alcohol percentage |
 ----------------------------------------------------
 | Red Keg Brewery | Saison     | 8,09               |
 ----------------------------------------------------

 Count: 1
BrewerSecond: Hmmm, interesting. That is a bigger value.
```

Figure 1: Output for Concurrency – solution 1



```
ConnectionTester: The thread 91 connected to the database.
ConnectionTester: The thread 92 connected to the database.
ConnectionTester: The thread 93 connected to the database.
ConnectionTester: The thread 94 connected to the database.
ConnectionTester: The thread 95 connected to the database.
ConnectionTester: The thread 96 connected to the database.
ConnectionTester: The thread 97 connected to the database.
ConnectionTester: The thread 98 connected to the database.
ConnectionTester: The thread 99 connected to the database.
ConnectionTester: The thread 1 closed connection to the database.
ConnectionTester: The thread 100 connected to the database.
ConnectionTester: The thread 2 closed connection to the database.
ConnectionTester: The thread 0 closed connection to the database.
ConnectionTester: The thread 3 closed connection to the database.
ConnectionTester: The thread 103 connected to the database.
ConnectionTester: The thread 11 closed connection to the database.
ConnectionTester: The thread 108 connected to the database.
ConnectionTester: The thread 107 connected to the database.
ConnectionTester: The thread 102 connected to the database.
ConnectionTester: The thread 109 connected to the database.
ConnectionTester: The thread 101 connected to the database.
ConnectionTester: The thread 110 connected to the database.
ConnectionTester: The thread 104 connected to the database.
ConnectionTester: The thread 111 connected to the database.
ConnectionTester: The thread 4 closed connection to the database.
ConnectionTester: The thread 8 closed connection to the database.
ConnectionTester: The thread 105 connected to the database.
ConnectionTester: The thread 106 connected to the database.
ConnectionTester: The thread 12 closed connection to the database.
ConnectionTester: The thread 112 connected to the database.
ConnectionTester: The thread 113 connected to the database.
ConnectionTester: The thread 114 connected to the database.
ConnectionTester: The thread 18 closed connection to the database.
ConnectionTester: The thread 5 closed connection to the database.
ConnectionTester: The thread 9 closed connection to the database.
ConnectionTester: The thread 10 closed connection to the database.
ConnectionTester: The thread 115 connected to the database.
ConnectionTester: The thread 117 connected to the database.
ConnectionTester: The thread 118 connected to the database.
ConnectionTester: The thread 23 closed connection to the database.
ConnectionTester: The thread 116 connected to the database.
ConnectionTester: The thread 119 connected to the database.
ConnectionTester: The thread 16 closed connection to the database.
ConnectionTester: The thread 13 closed connection to the database.
ConnectionTester: The thread 6 closed connection to the database.
ConnectionTester: The thread 7 closed connection to the database.
ConnectionTester: The thread 14 closed connection to the database.
ConnectionTester: The thread 15 closed connection to the database.
ConnectionTester: The thread 17 closed connection to the database.
ConnectionTester: The thread 19 closed connection to the database.
ConnectionTester: The thread 25 closed connection to the database.
ConnectionTester: The thread 20 closed connection to the database.
ConnectionTester: The thread 22 closed connection to the database.
ConnectionTester: The thread 21 closed connection to the database.
ConnectionTester: The thread 24 closed connection to the database.
ConnectionTester: The thread 26 closed connection to the database.
ConnectionTester: The thread 27 closed connection to the database.
ConnectionTester: The thread 28 closed connection to the database.
ConnectionTester: The thread 29 closed connection to the database.
ConnectionTester: The thread 30 closed connection to the database.
ConnectionTester: The thread 31 closed connection to the database.
ConnectionTester: The thread 32 closed connection to the database.
ConnectionTester: The thread 33 closed connection to the database.
```

Figure 2: Output for Output for Access points amount at the same time



```
BreweryOwner: I'm out of money and cannot pay my bills!
BreweryOwner: Let me change my brewery name in the database. I think that anyone finds my brewery then.
BreweryOwner: But first...
BreweryOwner: Cofee time!
DatabaseAdmin: One not paid order!
DatabaseAdmin: Right then, check the name of this industry
DatabaseAdmin: SELECT Brewery.name FROM Brewery WHERE id = 1997;
DatabaseAdmin:
 ------------------
 | Brewery name    |
 ------------------
 | Red Keg Brewery |
 ------------------

 Count: 1
DatabaseAdmin: There it is!
DatabaseAdmin: I should delete this company for its debt
DatabaseAdmin: First, time for a sip of a coffee
DatabaseAdmin: Cofee time!
BreweryOwner: It was really good.
BreweryOwner: Wonderful taste, now some work.
BreweryOwner: UPDATE Brewery SET name = 'Temporary Brewery Name' WHERE id = 1997;
BreweryOwner: Great job, my brewery is safe now.
DatabaseAdmin: It was really good.
DatabaseAdmin: Right, let's do it.
DatabaseAdmin: DELETE FROM Brewery WHERE name = 'Red Keg Brewery';
Microsoft.Data.SqlClient.SqlException (0x80131904): The DELETE statement conflicted with the REFERENCE constraint "FK
_Employee_Brewery". The conflict occurred in database "BreweriesDatabase", table "dbo.Employee", column 'brewery_id'.

The statement has been terminated.
   w Microsoft.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapClo
seInAction) w H:\tsaagent2\_work\11\s\src\Microsoft.Data.SqlClient\netfx\src\Microsoft\Data\SqlClient\SqlConnection.c
s:wiersz 2205
   w Microsoft.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean breakConnection, Action`1
 wrapCloseInAction) w H:\tsaagent2\_work\11\s\src\Microsoft.Data.SqlClient\netfx\src\Microsoft\Data\SqlClient\SqlInte
rnalConnection.cs:wiersz 773
   w Microsoft.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConn
ectionLock, Boolean asyncClose) w H:\tsaagent2\_work\11\s\src\Microsoft.Data.SqlClient\netfx\src\Microsoft\Data\SqlCl
ient\TdsParser.cs:wiersz 1668
   w Microsoft.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStre
am, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject stateObj, Boolean& dataReady) w H:\tsaagent2\_work\
11\s\src\Microsoft.Data.SqlClient\netfx\src\Microsoft\Data\SqlClient\TdsParser.cs:wiersz 2895
   w Microsoft.Data.SqlClient.SqlCommand.RunExecuteNonQueryTds(String methodName, Boolean async, Int32 timeout, Boole
an asyncWrite) w H:\tsaagent2\_work\11\s\src\Microsoft.Data.SqlClient\netfx\src\Microsoft\Data\SqlClient\SqlCommand.c
s:wiersz 3729
   w Microsoft.Data.SqlClient.SqlCommand.InternalExecuteNonQuery(TaskCompletionSource`1 completion, String methodName
, Boolean sendToPipe, Int32 timeout, Boolean& usedCache, Boolean asyncWrite, Boolean inRetry) w H:\tsaagent2\_work\11
\s\src\Microsoft.Data.SqlClient\netfx\src\Microsoft\Data\SqlClient\SqlCommand.cs:wiersz 1966
   w Microsoft.Data.SqlClient.SqlCommand.ExecuteNonQuery() w H:\tsaagent2\_work\11\s\src\Microsoft.Data.SqlClient\net
fx\src\Microsoft\Data\SqlClient\SqlCommand.cs:wiersz 1419
   w MultithreadedAccessTest.Database.MicrosoftSQLDatabase.Modify(String sqlStatement) w D:\Repositories\DatabaseOpti
mization\Report2\MultithreadedAccessTest\Database\MicrosoftSQLDatabase.cs:wiersz 36
ClientConnectionId:54e63c4b-906c-4fd5-88f7-4d53b559f3dc
Error Number:547,State:0,Class:16
DatabaseAdmin: It's done
DatabaseAdmin: Wait a minute!
DatabaseAdmin: There is no such brewery right now!
```

Figure 3: Output for Concurrency (solution 2)