

DbiUebung-001 -- Entity Framework

Todo:

Führen Sie die Tests mit `numberOfParallelThreads=1` aus. Verhält sich der Code wie erwartet?

Analysieren Sie den gegebenen Code auf Probleme. Wie manifestieren sich die Probleme bei den beiden Tests?

Beheben Sie die Nebenläufigkeitsprobleme mittels den gegebenen Locking-Arten und behandeln Sie jene Fehler, die dabei auftreten können.

- Zeilen-Locks
- Table-Lock
- Optimistisches Locking

- Bei nur einem Thread verhält sich der code wie erwartet, jedes query wird nacheinander abgearbeitet, bei mehreren parallelen Threads können jetzt bereits Probleme auftreten.

Ein Zeilen-Lock scheint deadlocks nicht zu verhindern und selbst mit dem lock-object im c# code, treten ab und zu deadlocks auf (je nach cpu auslastung)

- Ich fügte, wie bereits genannt "FOR UPDATE":

```
// Query with one resulting line
cmd.CommandText = "SELECT balance FROM konto WHERE kid=" + source + " FOR UPDATE"; // Line-Lock - for Update
```

- sowie eine Lock-variable `_locker` zum code hinzu:

```
#region FIELDS
private static readonly object _locker = new(); // lock-object to ensure save code execution
```

- Als letzte Maßnahme erweiterte ich den code um einen Vollständigen Table-lock, welcher die deadlocks letztendlich verhindert:

```
cmd.CommandText = "LOCK TABLE konto IN EXCLUSIVE MODE"; // Table lock - to prevent deadlocks
cmd.ExecuteNonQuery(); // while running multiple threads parallel
```

(auch bei vielen parallelen Threads, stimmen die Transaktionen und blockieren sich nun nichtmehr)



