DAT602 – Milestone 3

## Concurrency Management in MySQL vs .NET C#

## Alex Herbert

## 24/11/2024

#### Introduction

Concurrency management is the process that allows multiple operations to be executed in a database at the same time without conflicts or data inconsistencies. Both MySQL and .NET C# use distinct, individual methods to handle concurrency properly. Concurrency management is important when multiple transactions are being executed that interact with the same data simultaneously, this makes sure that transactions are not resulting in conflicting data or out of date transactions are being committed when the data has already been updated or changed.

This report focuses on the comparison between MySQL and .NET C# and how they handle concurrency, transactions, isolations levels and locking mechanisms and their implementation. In MySQL, concurrency is usually managed through transaction isolation levels and row-level locking, allowing for a more controlled and finely tuned approach to controlling the transactions being executed in the database. In .NET C#, concurrency management is handled using technologies such as Entity Framework and ADO.NET, offering both optimistic and pessimistic concurrency models.

#### MySQL Concurrency Management

MySQL implements transaction isolation levels and locking mechanisms to handle concurrency management, these methods allow for more control over how changes being made by a transaction are visible to other concurrent transactions.

###### Transaction Isolation Levels

**Read Uncommitted**

Select statements are performed in a nonlocking fashion, but a possible earlier version of a row might be used. Using this isolation level results in reads that are not consistent, also known as a dirty read. Otherwise, this isolation level works like read committed.

**Read Committed**

Each consistent read, even within the transaction, sets and reads its own fresh snapshot. This prevents dirty reads from occurring, but does allow non-repeatable read, resulting in data being able to be changed between reads within a transaction.

**Repeatable Read**

Consistent reads within the same transaction read the snapshot that is taken by the first read. This means that if several nonlocking select statements within the same transaction are consistent but also with respect to each other.

**Serialize**

This isolation level is like repeatable read but converts all nonlocking select statements to “SELECT … FOR SHARE” if auto commit is disabled. If autocommit is enabled, the select becomes its own transaction. It can then be a read only transaction and can be serialized if performed as a consistent read and does not block other transactions.

###### Pessimistic Locking in MySQL

This type of locking is used to prevent other users from performing operations or transactions on data tables or rows until a transaction is complete. This mechanism is built into the database, granting access to specific resources. This is generally used in scenarios where conflicts are the most likely and would cause data inconsistencies and errors. MySQL can implement row-level locking or table-level locking to stop updates to the same data from multiple transactions.

###### Optimistic Locking in MySQL

Optimistic Locking is not a feature in MySQL and relies on the applications logic to handle concurrency. This means that any changes to the data are allowed without implementing row-locking mechanisms, but the transactions validate the data before it is committed.

#### .NET C# Concurrency Management

Concurrency is .NET C# is used to handle multiple In-Out (I/O) operations, generally this is used for File I/O actions, network requests, and database queries in parallel to improve the performance of the application.

###### Concurrency using ADO.NET

ADO.NET uses transaction management with SqlTransaction or MySqlTransaction for concurrency management. The transactions comply with atomicity, consistency, isolation, and durability (ACID) properties to make sure data is consistent. In ADO.NET, transactions are controlled with the Connection object, this can then be initiated using the BeginTransaction method. Once the transaction has begun, a Command Object can be executed with the Transaction property. This can then commit, or rollback modifications made at the data source based on the success or failure of the transaction components.

###### Concurrency using Entity Framework

The Entity Framework uses an optimistic concurrency approach, which assumes that concurrency conflicts are relatively rare, which essentially means that there are no locks, but it arranges for the data modification to fail when it is saved if the data has been changed since the query was made. This approach normally reports that failure back to the application, which deals with the issue accordingly, either by retrying the operation again on the new data or throwing an exception, but that is dependent on the applications error handling logic. A Concurrency token is created, which is tracked when the database or entity is queried, then when and update or delete operation is performed, the value of the token on the database is compared against its original value read by the EF framework. This makes sure that the data that has been queried is consistent with the changes that have been made on the database, preventing inconsistencies in the data.

#### Conclusion

Concurrency management is crucial for making sure that consistency and performance is maintained in multi-user software applications. MySQL focuses primarily on isolation levels to make sure that transactions are not being conducted simultaneously on the same data, which essentially locks the table or row so that changes can’t be made by other transactions before it has been committed. The .NET C# approach to concurrency is through the use of frameworks such as ADO.NET or the Entity Framework, which uses methods to start transactions within the application logic, or by creating concurrency tokens that are checked to make sure that the data that is being queried is consistent with the changes that are being made. Choosing the correct approach to this depends on the type of project, the applications architecture and workload. For database intensive operations, MySQL transactions offer a robust and reliable method for concurrency management whilst the Entity Framework’s optimistic approach is generally used for application driven logic.

## Citations and References

Microsoft. 2024. Handling Concurrency Conflicts.

<https://learn.microsoft.com/en-us/ef/core/saving/concurrency?tabs=data-annotations>

Microsoft. 2024. Local Transactions.

<https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/local-transactions>

MySQL. 2024. 17.7.2.1 Transaction Isolation Levels.

<https://dev.mysql.com/doc/refman/8.4/en/innodb-transaction-isolation-levels.html>