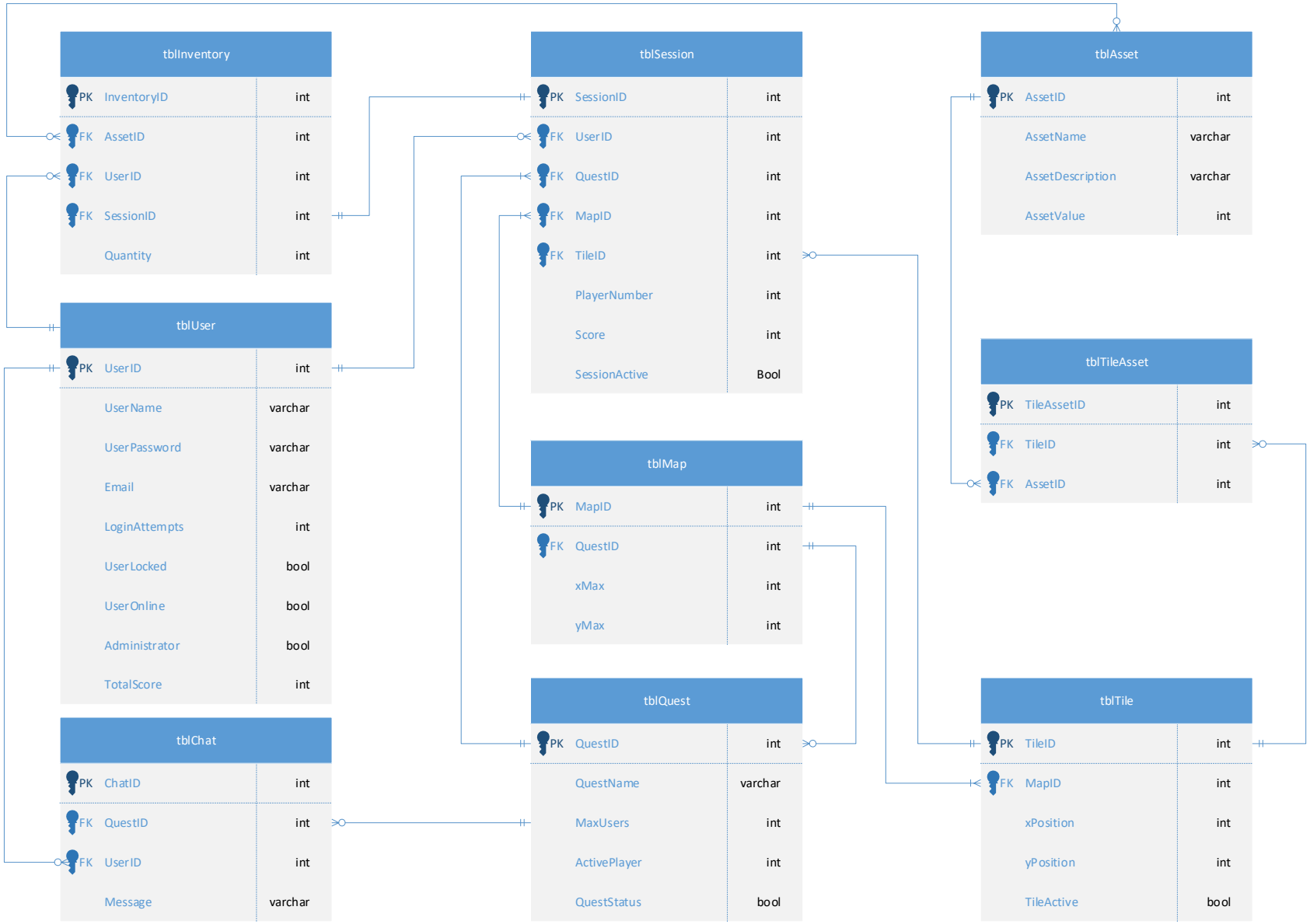
# DAT602 – ASSESSMENT TWO
# PROJECT
# Milestone Two

Daniel Waller : 158 758 567

# Table of Contents

# Entity Relationship Diagram – Iteration Two




Daniel Waller : 158 758 567

# CRUD Table – Iteration Two

| Entity/Attribute | User Registers | User Login | Delete Account | Lock Account | Join Quest | New Quest | User Moves | User Chat | User Leaves Quest | Quest Ends | User Logout | Kill in Progress Quest | Delete User | Create User | Modify User |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **tlbUser** | C | R | RD | | R | R | R | R | R | RU | RU | | RD | RC | RU |
| UserID | C | | D | R | R | R | R | R | R | R | R | | RD | C | R |
| UserName | C | R | RD | | | | | R | | R | | | D | RC | RU |
| UserPassword | C | R | RD | | | | | | | | | | D | C | RU |
| Email | C | | D | | | | | | | | | | D | C | RU |
| LoginAttempts | C | RU | D | | | | | | | | | | D | C | RU |
| UserLocked | C | RU | D | RU | | | | | | | | | D | C | RU |
| UserOnline | C | RU | D | | | | | | | | RU | | D | C | |
| Administrator | C | R | D | | | | | | | | | | D | C | RU |
| TotalScore | C | | D | | | | | | | RU | | | D | C | RU |
| **tblQuest** | | | | | R | C | RU | R | RU | RD | | RD | | | |
| QuestID | | | | | R | C | R | R | R | RD | | RD | | | |
| QuestName | | | | | | RC | | | | RD | | D | | | |
| MaxUsers | | | | | | C | | | | RD | | D | | | |
| ActivePlayer | | | | | | C | RU | | RU | RD | | D | | | |
| QuestStatus | | | | | | C | | | | RD | | D | | | |
| **tblSession** | | | D | | RC | C | RU | | R | RD | | RD | RD | | |
| SessionID | | | D | | RC | C | R | | R | RD | | D | D | | |
| UserID | | | D | | RC | C | R | | R | RD | | D | RD | | |
| QuestID | | | D | | R | C | R | | R | RD | | RD | D | | |
| MapID | | | D | | R | C | R | | R | RD | | D | D | | |
| TileID | | | D | | C | C | R | | R | RD | | D | RD | | |
| PlayerNumber | | | D | | C | C | R | | | D | | D | D | | |
| Score | | | D | | C | C | RU | | | RD | | D | D | | |
| SessionActive | | | D | | C | C | | | RU | D | | D | D | | |
| **tblInventory** | | | D | | C | C | RU | | | RD | | D | RD | | |
| InventoryID | | | D | | C | C | R | | | RD | | D | D | | |
| AssetID | | | D | | C | C | R | | | D | | D | D | | |
| UserID | | | D | | C | C | R | | | RD | | D | RD | | |
| SessionID | | | D | | C | C | R | | | RD | | D | D | | |
| Quanity | | | D | | C | C | RU | | | D | | D | D | | |
| **tblChat** | | | D | | | | | C | | RD | | D | RD | | |
| ChatID | | | D | | | | | C | | D | | D | D | | |
| QuestID | | | D | | | | | C | | RD | | D | D | | |
| UserID | | | D | | | | | C | | D | | D | RD | | |
| Message | | | D | | | | | C | | D | | D | D | | |
| **tblMap** | | | | | R | C | R | | R | RD | | D | | | |
| MapID | | | | | R | C | R | | R | RD | | D | | | |
| QuestID | | | | | R | C | R | | R | RD | | RD | | | |
| xMax | | | | | | C | | | | D | | D | | | |
| yMax | | | | | | C | | | | D | | D | | | |

| Entity/Attribute | User Registers | User Login | Delete Account | Lock Account | Join Quest | New Quest | User Moves | User Chat | User Leaves Quest | Quest Ends | User Logout | Kill in Progress Quest | Delete User | Create User | Modify User |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **tblTile** | | | | | R | C | RU | | RU | RD | | D | RU | | |
| TileID | | | | | R | C | R | | R | RD | | D | R | | |
| MapID | | | | | R | C | R | | R | RD | | D | | | |
| xPosition | | | | | R | C | R | | | D | | D | | | |
| yPosition | | | | | R | C | R | | | D | | D | | | |
| TileActive | | | | | R | C | RU | | RU | D | | D | RU | | |
| **tblTileAsset** | | | | | | C | RD | | | RD | | D | | | |
| TileAssetID | | | | | | C | RD | | | RD | | D | | | |
| TileID | | | | | | C | RD | | | RD | | D | | | |
| AssetID | | | | | | C | RD | | | RD | | D | | | |
| **tblAsset** | | | | | | R | R | | | | | | | | |
| AssetID | | | | | | R | R | | | | | | | | |
| AssetName | | | | | | R | R | | | | | | | | |
| AssetDescription | | | | | | R | R | | | | | | | | |
| AssetValue | | | | | | R | R | | | | | | | | |

## ERD and CRUD Iteration Two Discussion

Minor changes to ERD and CRUD have been made based on feedback and learning from creating MySQL procedures during milestone two development. Specific changes include:

- Minor changes to ERD relationship notations based on feedback.
- Additional SessionActive attribute to tblSession to indicate if a player has left the quest.
- The removal of the scoring table and the addition of a Score attribute in tblSession.
- Refactor the chat table to have both the QuestID and UserID as foreign keys.
- Minor changes to table names for shorter and terms and better understanding when coding.

## CURD Activities MySQL Procedures

Each CRUD activity has been developed into a MySQL procedure and tested in the console application provided with this document.

### User Registration

See the userRegistration procedure that creates a user record in tblUser, respectively.

### User Login

See the userLogin procedure that verifies user credentials and login activity.

### Delete Account

See the userDelete procedure that verifies the user credentials and deletes the user records.

### Lock Account
Included as part of the login procedure.

### Join Quest
See the joinQuest procedure that verifies the chosen quest and inserts the user's details into the game if successful.

### New Quest
See the newQuest procedure that verifies chosen quest name and builds a new quest, map, tiles, assets, and user details.

### User Moves
See the userMove procedure that verifies the user's input, scores point, update tiles, and inventory.

### User Leaves Quest
See the leaveQuest procedure that updates the user's quest status and ensure the tile they were on is now available for other players.

### Quest Ends
See the questCheck procedure that runs at the end of each userMove procedure. It validates the current quest to see if there are any players with a winning score and that there are still assets available. If the game is over, the questCheck procedure ends the game, updates the user's total score and informs them of the winner.

### User Logout
See the userLogout procedure that updates online status.

### Kill In Progress Quest
See the administratorKill procedure that kills any in-progress quest and removes all records.

### Delete User
See administratorDelete procedure that deletes selected user and their records.

### Create User
See administratorAdd that creates new user records.

### Modify User
See administratorModify that modifies existing user records.

# ACID and Multi-Player Game Support

ACID, according to Jepson (2001), is an excellent way to test overall quality. ACID is an acronym that describes four properties of a robust database system: atomicity, consistency, isolation, and durability. These features are scoped to a transaction, a unit of work that the programmer can define. A transaction can combine one or more database operations.

## Atomicity

Jepson (2001) states that atomicity is an all-or-none proposition. Suppose you define a transaction that contains an UPDATE, an INSERT, and a DELETE statement. These statements are treated as a single unit with atomicity, and thanks to consistency (the C in ACID), there are only two possible outcomes: either they all change the database, or none of them do. Furthermore, Kaur (2021) states that atomicity is also known as the 'All or nothing rule', meaning transactions must be executed in their entirety to ensure the correctness of the database state.

With Wizard Quest, the procedures are designed to complete single functions for users. If the process does not complete, the database will not be updated. For example, the userMove procedure handles a player moving, scoring any points associated with that move, and setting the following players turn all in the same procedure. If the database failed at any point during the move, none of these changes would happen, keeping the database in a stable state. Additionally, the game structure is designed to ensure only one player at a time can move in a quest. This adds a further constraint to the database, ensuring only one user is making changes in a game.

## Consistency

Consistency links into the previous discussion about users moving in the game, and Jepson (2001) discusses, consistency guarantees that a transaction never leaves your database in a half-finished state. If one part of the transaction fails, all of the pending changes are rolled back, leaving the database as it was before you initiated the transaction. For instance, when you delete a customer record, you should also delete all of that customer's records from associated tables.

In Wizard Quest, all quest data is deleted, and players scores are calculated when the quest ends, including any map entries, assets, and user quest details. If the game ends successfully, MariaDB (2018) states, new data will be added to the database and the resulting state will be consistent with existing rules.

## Isolation

Jepson (2001) discusses isolation as keeping transactions separated from each other until they're finished. Furthermore, Tech School (2020) explains that when working with database transactions, one crucial thing we must do is choose the appropriate isolation level.

- The lowest isolation level is **read uncommitted**. Transactions in this level can see data written by other uncommitted transactions, thus allowing dirty read phenomenon to happen.
- The next isolation level is **read committed**, where transactions can only see data that other transactions have committed. Because of this, dirty read is no longer possible.
- A bit more strict is the **repeatable read** isolation level. It ensures that the same select query will always return the same result, no matter how many times it is executed, even if some other concurrent transactions have committed new changes that satisfy the query.

- The highest isolation level is **serializable**. Concurrent transactions running in this level are guaranteed to yield the same result as if they're executed sequentially in some order, one after another without overlapping.

Importantly Oracle (2021) state that isolation level read uncommitted and serializable change processing behaviour so drastically that they are rarely used.

Given these insights and further discussions in class, a global isolation level of read committed has been applied to the Wizard Quest database. Isolation at this level protects the integrity within each transaction and prevents dirty reads. This level, combined with the nature of the game being a turn-based game, is the best solution ensuring each transaction is completed with the latest data available, system performance is maintained, and accurately supports a multiplayer transaction environment.

## Durability

Jepson (2001) discusses that durability guarantees that the database will keep track of pending changes in such a way that the server can recover from an abnormal termination. Hence, even if the database server is unplugged in the middle of a transaction, it will return to a consistent state when it's restarted. Furthermore, Kaur (2021) state durability ensures that once the transaction has completed execution, the updates and modifications to the database are stored in and written to disk, and they persist even if a system failure occurs. These updates now become permanent and are stored in non-volatile memory. The effects of the transaction, thus, are never lost.

The ACID properties, in totality, provide a mechanism to ensure correctness and consistency of a database in a way such that each transaction is a group of operations that acts as a single unit, produces consistent results, acts in isolation from other operations and updates that it makes are durably stored.

# References

Jepson, B. (2001). *Building better databases.* Web Techniques. https://people.apache.org/~jim/NewArchitect/webtech/2001/09/jepson/index.html

Kaur, A. (2021). *ACID Properties in DBMS.* Geeks for Geeks. https://www.geeksforgeeks.org/acid-properties-in-dbms/

MariaDB. (2018). *ACID Compliance: What It Means and Why You Should Care.* MariaDB https://mariadb.com/resources/blog/acid-compliance-what-it-means-and-why-you-should-care/

Oracle. (2021). *MySQL 8.0 Reference Manual*. MySQL. https://dev.mysql.com/doc/refman/8.0/en/glossary.html#glos_isolation_level

Tech School. (2020). *Deeply understand Isolation levels and Read phenomena in MySQL & PostgreSQL.* Tech School Guru. https://dev.to/techschoolguru/understand-isolation-levels-read-phenomena-in-mysql-postgres-c2e