DAT602 – Milestone 2

Mark Christison

Nelson Marlborough Institute of Technology, New Zealand

# Contents

[Contents 2](#_Toc42068626)

[Event Table 3](#_Toc42068627)

[References 4](#_Toc42068628)

# Introduction

This document carries on the work done previously in Milestone 1 where the business rules, storyboards and an outline for the procedures need to be created where laid out. This milestone revolved around completing the SQL required for making the database schema and the procedures needed for the game play. To ensure the correctness of the database and functionality of the build, a set of test data has been included in the SQL. This serves functionally as it provides the initial game dataset.

After the creation of the database was completed, a C# console application was created to interface with the database and test its functionality. The app has been written to accept dynamic input as to allow for complete testing of the database and to catch any and all possible errors in the code.

Additionally, as transactions are being used to interact with the database, note has been taken to ensure that those transactions have atomicity, consistency, isolation, durability (ACID) in the case of any errors or disasters occurring to the database.

M**ILESTONE TWO - Implement a command line test in the selected programming language for the game using DML statements in DB procedures/transactions.**

**Milestone two is made up of 60 marks worth 30% of the total marks.**

For milestone two, hand-in all code you have written in SQL and Visual Studio,  write a single report that describes what you implemented and the rationale for the choices you made as you undertook the following:

* From your previous analysis and design Identify each activity and develop SQL PROCEDURES (and TRANSACTIONS) for all of your game’s activities. Develop a set of test SQL PROCEDURES that demonstrate the running of games. Including such things as
  + Live game play, [4]
  + Player registration, [4]
  + Player selection [4]
  + Confirmation for a game [4]
  + Game administration functions. [4]

         (20 marks)

* Describe how your system supports multi-player game play, in terms of transactions and database table specification in the target DBMS. Discuss how the transactions are reliable in terms of the acronym ACID:
  + Atomicity [4]
  + Consistency [4]
  + Isolation [4]
  + Durability [4]

         (16 marks)

* Create a suitable set of test data for the whole game.   
  Include all cases identified in your CRUD analysis.  
  Marks are deducted for errors.(10 marks)
* In Visual Studio (CLR).net , implement a CLASS called Test in your game’s namespace that runs all the procedures and functions you defined in SQL. (Checks connection to SQL and SQL)

        Marks are deducted for errors. (10 marks)

* In a Main method for your command line test application, write .net code that  instantiates your Test class, and calls all of the methods with suitable parameters.

        Marks are deducted for errors. (Checks they get called);

         (4 marks).

Include in your report, how you relate your selection of test data to your earlier analysis, and sketches of interactions depicted in your earlier storyboards. Refine your understanding of the scenarios of the game. Identify specific scenarios from your design. Refine, develop and refactor your SQL to meet these. Marks are given for describing your rationale and processes, as well as plain descriptions of the database and application you are developing.

# SQL

## DDL

In creating the procedures needed for the usages scenarios I have realised that the database requires a few changes. Most of the changes have happened to the primary keys on several tables DDL. While initially manually incrementing each data row that was entered to the table was fine, I knew that this needed to be smarter for adding extra data especially in game play. While it was an initial consideration to give these tables an ID column that was an integer, I have instead opted of more natural keys in the data.

The user table has a natural key in the username. Understandably, this might slow down the database if there were to be a large number of rows in the table, however, the size of the database for this problem, it should not grow large enough for that to become an issue. However, I do note it as a potential issue in the future that may require changes.

As with the user table, the skills table, item table, mine table, map table, and character table all have used the attribute that “names” them as a primary key.

An auto incremented integer primary key has been added to the chat table as there was no other logical way to reference every row from the data in the table already.

The other changes to the DDL include the addition of “On update cascade” and “on delete cascade” to the foreign key columns in several tables. This is to support the changes that could occur to the data across the database and to not orphan any data. An example of this is in table character on the column username. As one of the usage scenario’s is for a user changing their name, if the username column was not updated when the primary key in the user table was, then the data rows with the old user name would no longer reference the new user and all old data would be orphaned. Likewise, if a user deletes their account, all the characters that user had will be deleted as they are no longer needed by the database.

# Inserted Test data

To meet the new DDL, the test data has been updated. New test data has been included as well to more accurately reflect the game that is being created. All the concepts from the draft in milestone 1 have been fleshed out in the database. All game assets have been created and a set of different maps has been created.

The inserted data ensures that the tables still have the initial integrity that they had during the tests carried out for milestone 1.

## Update Statements

The initial update statements where intended as tests only and have therefore been commented out. The newly created procedures/transactions now encompass all the required updates for the game.

## Delete Statements

The delete statements have been removed as the created database now has a functional set of data and the deletion of the data would disable some of the functionality, such as the administrator procedures. As with the update statements, these have been commented out.

## Procedures/Transactions

The original usage scenarios as defined in milestone 1 have provided a template for constructing the procedures. Notable, several usage scenarios have been reduced from 2 or 3 scenarios to 1 procedure. As well, other newly discovered scenarios have been created into procedures that the game will require. Some of the scenarios have been split into 2 procedures to allow for user input as needed.

### Register a User

SQL Lines 400-424.

This procedure checks to see if the username or email passed as parameters already exist in the database. If not then this procedure inserts the parameters into the user table. It also catches exception cases where a username or email are already in the database.

### User Log in

SQL Lines 428-480.

This procedure checks for the username and password parameters being in the same row, if they are it updates that data row’s column isOnline to true.

### User Log off

SQL lines 484-503

Refers to storyboard 13 From the character selection screen, a user can choose to modify their account.

## Edit User

SQL lines 507 – 543

## Delete User

SQL lines

Refers to storyboard 15. From the character selection screen, a user can select one of of their characters and choose to delete it.

User Locked out

Refers to storyboard 10. If a user has 5 or more failed log in attempts, they are ‘locked out’ of their account and require an administrator to unlock their account.

User Creates Character

Refers to storyboard 4. A user creates a new character that is associated with their user account.

User Deletes Character

Refers to storyboard 14. A user chooses to delete their character associated with their user account.

Select Character to play game

Refers to storyboard 5. The logged in user selects from a list of characters to play the game with.

Create Game

Refers to storyboard 6. Once the user is logged in and has selected character they can choose or be chosen to play a game.

Leave Game

Refers to storyboard 9. During game play a character can leave the game, saving the state of the character in the game and returning to the game menu.

Game Generates Items

Refers to storyboard 8. When the game is generated, the items inside the game are also generated.

Game Generates Mines

Refers to storyboard 8. When the game is generated, the mines inside the game are also generated.

Return to Game in Progress

Refers to storyboard 16. From the game menu when a player has left a game in play they can return to that game

Character makes Move

Refers to storyboard 9. During game play, a character clicks or presses a key associated with a movement to another tile.

Character picks up item

Refers to storyboard 9. During gameplay, if a character is standing on a tile with an item and they take the appropriate action to pick up the item (possibly a key press) it is added to their inventory.

Player Uses Item on Mine

Refers to storyboard 9. During gameplay, if a character has an item that matches a ‘mine’(place to use items) and the matching skill, they can ‘use’ the item (with a keypress). Using the item results in an increase in points and a decrease in the item’s durability.

Character makes a chat

Refers to storyboards 6, 8, 9, and 11. Once a character has been chosen for game play, in gameplay or in the admin panel, a character can chat with all characters.

Access Administrator Interface

Refers to storyboard 11. A character who is an admin can from the game menu access an administration window where they have administration rights over the games, characters and users.

Admin Kills Running Game

Refers to storyboard 11. An administrator can kill any/all running games.

Admin adds a user

Refers to storyboard 11. An administration can add users to the database.

Admin edits user

Refers to storyboard 11. An administrator can modify any user account in the database.

Admin deletes user

Refers to storyboard 11. An administrator can delete any user account in the database

Admin unlocks locked user

Refers to storyboard 11. An administrator can unlock any currently locked user accounts.

# Test Data

# ACID

# References