Journals- DAT602 Class

By: Abhimanu Saharan
NMIT ID: 1349714

Week 1

No class

Week 2

This week we looked at the course outline- project requirements and the timeline.
The project is a single player tile based game (i.e- no two players play using the same device, but play against each other), and the state of the game is to be saved even if the player leaves the game. The project is divided into three milestones:

Milestone 1- Hand in of Database Design (CRUD Tables), Storyboards, Usage Stories and Use cases.

Milestone 2: Connect Database to CLI, and test database procedures.

Milestone 3: Create prototype graphical interface for game and make any updates if required.

This week I started by creating the CRUD Tables. At the time of writing this journal, I'm thinking how I'd normalize these many tables. ERD diagrams would form the base of this game, so I should spend time on that, instead of editing and figuring out the table's relationships later.

Also in this first class we installed MySQL and set up Visual studio to write SQL scripts.

This week we also looked at an example of a procedure in SQL.

Week 3

This week we practised some more SQL examples (converting an Access database to MySQL) and learned more about SQL statements like JOIN, ALTER and DELETE.

I went through the SQL tutorial on W3Schools, where I refreshed my SQL knowledge and tried some examples. I was making some syntax errors like using double quotes instead of single quotes/using quotes at places where I did not have to. The tutorial gave me a basic understanding of SQL. Next I will be practising using SQL procedures to manipulate data in other tables based on different combined conditions.

Week 4

This week I looked at stored procedures in MySQL. Stored procedures look similar to functions in C# or other OOP languages, and also the way they take parameters using in, out and inout keywords before variable name looks similar to reference and value parameters in C.

I also looked at conditions and loops in MySQL. I could not see much difference in the working of LOOP and REPEAT, except the use of different keywords to stop their execution (LEAVE using 'ITERATE and LEAVE' and REPEAT using UNTIL), but hopefully the differences will start getting clearer as I start using them.

In class we looked at flow control diagrams for Login using SQL procedures.

I think the flow control diagram would be helpful for the whole game, because the determination of different procedures would be better understood and analysed using flow control diagrams.

Week 5

This week we looked at DELETE ON CASCADE and deleting data in multiple tables using JOINS.

I'd be using 'Delete On Cascade' to remove the players from the database and all their existing data in other tables would be automatically removed, and many more functions. Similarly I'd be using 'On cascade' with 'Update' to update the data on gameplay in related tables.

Week 6

This week I connected MySQL to C# and things are getting very clearer from here. I used the example provided in class, and then referred to documentation of ADO.net. The documentation is too intense for me or it's just that my understanding of OOP is not very clear. I'll stick to using DataAdapter to both retrieve and update data. In the documentation in some places DataReader is used to retrieve data and 'SqlCommand' is used to update data, and then the Objects like DataSet (another name for 'table'), DataColumn are used which makes it even more confusing.

All in all, now I know how to connect SQL server to C#.Net. This is very much similar to how I connected to MongoDB with express in my Web Development class- by creating an instance of the database connection, and then fetching data from that instance.

Week 7

In class we looked at and practised using 'Group By' and 'Having' clauses in SQL, and continued working on creating DML procedures for our game. (Even though at this point, because of my bad habit of procrastination I have lost track with the rest of class and have only submitted Storyboards and basic DML- as part of Milestone 1)

Week 8

This week I learnt about user defined functions, procedures and views.

From what I have learnt the following is my hypothesis, which will be tested when I start using functions and procedures extensively for my game:

Views: Virtual tables, used to simplify complex/long queries or to hide some columns.

Functions: For returning a value after working on/calculating the required logic from the inputs. Can be used in SELECT, WHERE, HAVING etc clauses, therefore- can be used to do routine work over the passed data, eg in our case - a function can be called to check if a user is logged in, instead of writing queries every time before the tasks like stop game or update password are performed.

Procedures- Everything that functions can do but they don't need to return values, and procedures can modify the database which functions can not.

Week 9

This week we looked and practiced using subqueries, ‘On delete cascade’ and ‘On update cascade’.

We looked at examples __

Week 10

This week I learned about transactions in SQL. Transaction is based on the concept of ACID(Atomicity, Consistency, Isolation and Durability),- opposite/unlike of BASE(Basically Available, Soft State and Eventually Consistent) which is used in many modern nonSQL databases, which has its own benefits (keeping track of data using nodes which are not saved yet but will be saved eventually)

Transactions are used with the ‘commit’ keyword to make the changes made in the transaction permanent. The changes can be cancelled (if not committed) by using ‘rollback’ keyword.
Transactions are used mostly in try catch blocks, where if an error is caught the transactions could be rolled back.

The ‘savepoint’ keyword can be used in a transaction to specify a point, from after which the transaction will be rolled back using rollback keyword.
The transaction would be very beneficial in the game database as I could use it in try catch blocks for checking that all the related fields (eg- score and high score simultaneously) have been updated, or else the transaction could be rolled back to a certain point, and in many other cases.

This week we also practised connecting our database to CLI in C#.Net. This was done in the following way:

1. Create an instance of MySqlConnection and pass the connection string into that.
2. SQL procedures are called using MySqlHelper.ExecuteDataset and passing the following into it- the created instance of MySqlConnection, Procedure Name (with parameter names-if any), array of parameters- if any.
3. MySqlHelper returns a dataset. That dataset is a collection of tables which in turn contains DataRow and DataColumn. Therefore a particular column in returned value can be accessed by datasetVariableName.Tables[0].Rows[0][‘columnName’].

Week 11

This week we learned about isolation levels in MySql. The isolation levels are used to control the access of transaction data before it is committed or rolled back. Eg:

Setting the isolation level to ‘Read uncommitted’ allows the data in the transaction to be read even if the transaction has not completed. ‘Read committed’ does not allow the uncommitted data to be read.
The other two isolation levels are :
‘Repeatable Read’- stops the other transactions from modifying the data that is being used by some transaction that is not yet committed.
‘Serializable Read’ - is a more restrictive version of ‘repeatable read’ which does not allow even the modification of data that is in the range of data currently being read by any transaction.

I think I’ll most of the time be sticking to the default isolation level where the uncommitted data could not be read by other connections , but I would find serializable handy if I want to restrict updating the data related to my transaction before it is committed. Eg: Player high score data should not be allowed to update before the player exits the game.

We also learned about ‘locking reads’ in MySql where a ‘Select..for share’ or ‘select..for update’ can be used to prevent reading or editing of data respectively that is being read by the select statements. __

Week 12

This week we checked out LINQ in C#. Linq enables converting a Relational database to a collection of objects and using LINQ methods to query and update that data. I find the feature amazing, but I’ll stick to using my normal SQL queries and just a few LINQ commands, because of the following two reasons:

1. At the time of writing this journal, it is the last day of submission. Therefore I don’t want to spend time learning the LINQ way of querying data.
2. The marking for our project requires more SQL and less data manipulation using LINQ. __

Week 13

This week we practised creating DataGrids in .Net, updating and retrieving data.
The Datagrid has a DataSource property which links the grid to the Datasource.
To convert a database to an object DataAdapter class is used.
Datasets are collection of tables, which then contain Table properties, which contains __

Week 14

When the game starts the login form is loaded: which, on entering the username and password queries the player table

1. If the player exists, it checks for the password field and returns the result.
   If result is false the user is not logged in and error is displayed.
2. If player does not exist:
   The player ta

```
Procedure checkLogin(){
if (pusername=player.username){
if(ppassword
```