Game Studio Database

Student Name: Richard Whitney

Student Number: 20040645

Date: 30/12/18

Module: Database

Implementation

I started the process by writing a business description for a game studio. While writing the document I tried to keep in mind what functions I wanted the database to preform. Once I completed the document I then went through it and picked out entities that I thought where relevant. I created an ER diagram with the basic relations between each entity. I found this part of the project to be one of the hardest. I had to revise the diagram multiple times before I could implement the database in MySQL. I think this was the most important part of the project as once I had an Enhanced ER diagram that I was happy with a good part of the implementation when smoothly. Next I created the normalised tables from the diagram, including the primary keys and foreign keys where needed. Finally I outlined a data dictionary for each table in the database.

The first step of the implementation I took in MySQL was the create all the tables in the database. This was fairly straightforward due to the work done in the design phase of the project. I did change the types of the columns in some tables and increased the size of some as well. I then began to populate all the tables. I originally inserted 10 games into the Game table but reduces it to 3 games when I went on to populate the WorksOn table. This was because the combination of employees and games led to the need for a lot of entries in this table. I felt I had to reduce the number of games to make this part of the implementation more manageable.

The main problem that I encountered during this phase was what way I should handle how entries are added into the Platform and PC tables. The PC table is a sub-table of the Platform table so I needed a way to keep the tables consistent with each other. I tried to solve this by using a trigger that would add a record to the Platform table whenever a record to the PC table was added but I couldn’t get this solution to work correctly. The solution I settled on was to include an additional column in the Platform table called type, used to determine if a game is on PC. I then created a procedure call insertPlatform that automatically adds an entry to the PC table if the entry has the type “PC”. I then defined the users for the database and defined their privileges.

The final part of the implementation was to create the list of frequently used queries for the database. I already had a good idea of the queries I wanted to use from the work done in the design stage of the project. There was two areas that I found the most difficult during this part of the project. The first was to display the number of people working on a game during each phase of its development. My first attempts to tackle this problem used sub-queries but I couldn’t retrieve the correct data. The solution I came up with was to create a view with the relevant records I required and then use a simpler query on the new view which worked. The second query that I had difficulty with list how long each game took to develop. As before I initially used sub-queries but settled on creating a new view with the sub-set of records that I needed. I then queried the view, however the query is still not functioning 100% correctly. There is a bug where if a game is still in development it will show as having spent 0 days in development. I tried to correct this using an if statement in the ‘datediff()’ part of the query but I couldn’t get it to function correctly.

Frequently Used Queries

1. Select all games that an employee has worked on:
   Useful for the studio to know what games an employee has worked on. This could be used in determining performance metrics and assigning bonuses.
2. Select all employees that worked on a certain game:
   Similar to the first query. This could be used in assigning bonuses based on a games success. The studio could also use this data to determine which employees work well together.
3. Select all games worked on during a certain time period:
   The studio could use this to help determine how busy is was during a certain time period.
4. Select all employees and their current projects (games):
   The studio could use this to keep track of whose working on the current game. Also will show if an employee dose not currently have a project.
5. Select all employees and their current roles:
   Similar to the last query. This gives information on what role every employee has on the current project.
6. Select all employees and the department that they currently work in:
   Useful for the studio to know what department every employee belongs to.
7. Select all employees that worked on a certain game and their roles during the development:
   Displays the role of all the employees that worked on a certain game, with the start-date and end-date of the their roles. An employee can have different roles during the development of a game. This will help the studio know what employee worked on what roles.
8. Select the current game in development, the current phase it is in and the date it entered the phase:
   Useful for the studio to know what game is currently in development and at what stage of development it is in
9. Select the number of employees working on a certain game during each phase of its development:
   Help the studio to manage employees more efficiently. The studio can see how many employees are working on each game at each stage of its development. The studio could reassign people to different games based on this information.
10. List how long each game was in development:
    Allows studio to preform an analysis on past development.