Final Report

THE LEAGUE GENIE

Analysis of The League Genie game predictor

Madison Bruner

Sasha Chen

5/22/15

Table of Contents

Executive Summary

This document begins with a description of the initial problem that led to this project's inception. The specific solution to that problem, the major challenges of the solution, and the overall design of the database used are discussed following the introduction of the problem. Finally an analysis of the results of implementing the solution is examined, citing both the strengths and weaknesses of the solution. The final entity relationship diagram and relational schema diagram for the project database are provided at the end of this document.

Introduction

This document is the final report for The League Genie (TLG) created by Madison Bruner and Sasha Chen. Its purpose is to analyze the problem outlined in the problem statement and discuss the solution by our team. In examining our solution, this report will also discuss the strengths and weaknesses of the approach as well as revisit the Security Analysis and Final Problem Statement documents.

Problem Description

League of Legends (LoL) is one of the most popular online games in the world with over 70 million accounts and more than 12 million unique players logging on every day. LoL is a very competitive game for most people and winning really matters to them. Naturally people want to know their chances of winning games at various points in the game. Our project is to create a database that will take in the current game state at a certain time and calculate who the projected winner of the game will be.

Our goal in creating TLG was to give players a means to see how they are doing in a particular game. If their projected outcome is to lose, it may encourage them to play more carefully or try a new strategy. If their projected outcome is to win, it may encourage them to keep doing what they are doing and reinforce their confidence in their play style.

Below is a table of features that the finished version of TLG will include

Feature Name	Feature Explanation
Desktop application	The system will be accessible through a java based desktop application.
Average database summoner level	User can see the average level of the players in the database to get a sense of the average level of experience of those players.
Champion search	User can search for all matches associated with an inputted champion name.
Post-game statistics	After inputting a game id, post-game statistics associated with it can be viewed.
Administrator interface	Interface allows administrators to more easily add data to TLG database.
Game outcome prediction	Application is able to predict the outcome of a game given its information from the user.

Solution Description

The solution that was decided upon for TLG involved using Java and Microsoft SQL Server in conjunction to create a GUI for the user to interact with and to store all of the data for TLG respectively.

Front-end

On the front-end of the application, it was attempted to make input of information as simple as possible. Although the user must input as much information as they can. This leads to the fact that there needs to be many text boxes for the user to input information for the main game prediction feature of TLG. The interface is simple, with labeling for what should be inputted where. It includes an array of text boxes for the user to input information about their game as well as a couple text boxes for the champion search and post-game stats features. There are also buttons associated with each input set to execute the procedure linked to the text boxes that are to be filled in by the user. Also available on the front-end is an administrator interface. This interface is only accessible to users with the proper admin credentials. Much like the normal user interface, the administrator interface also has a large array of text boxes to input information into the database. Unlike the normal user interface, this information actually gets entered into the database rather than just being compared to information in the database.

Back-end

The GUI that the user interacts with uses TLG database on one of Rose-Hulman's servers as its host of information. Champions, Items, gold, kill, death values and more are located in this database. Information in the database is read-only for normal users in order to add an extra level security for data integrity. A more detailed description of TLG database is located in the Database Design section.

Challenges Faced

Challenge: Only two people to work on the entire application.

Solution: The work for TLG was mainly split into three sections: front-end, back-end, and documentation. Our team decided to split the work up with respect to these categories with Sasha Chen mainly implementing the front-end GUI and Madison Bruner being responsible for most of the back-end work and documentation.

Analysis: This approach worked magnificently. Because there was a bit more work to be done on the front-end than on the back-end, it worked out well that the front-end was done by one person and the back-end and documentation was done by the other. While both members were responsible for their own parts, there was still heavy collaboration between them on those parts.

Challenge: Java isn't the friendliest language for implementing a GUI.

Solution: Lots of internet searching and deciphering of Stack Overflow questions and answers.

Analysis: Although it took a great amount of time to search for relevant and helpful information on the internet. In the end, a solution was to the particular problem that Java was having was always reached.

Challenge: Front-end and back-end features were heavily dependent on each other and it was difficult to do one before the other.

Solution: When it was decided what a feature would be, Madison Bruner and Sasha Chen would discuss together exactly what inputs would be needed for a procedure and exactly what outputs it would give back to the front-end.

Analysis: This approach worked great most of the time because of the pre-planning that was done. There were a few instances where arguments needed to be re-ordered so that they would be easier for the user to input on the front-end, but for the most part this was not needed.

Database Design

For the reader's reference, an Entity Relationship Diagram and a Relational Schema are provided at the end of this document.

Database Security

For normal users, the database is read-only. This is the first layer of security for the database. By disallowing normal users to change tables in the database, they are not even given the opportunity to enter bad data for accidental or malicious reasons. To make changes to the database, a user must have administrator privileges (currently limited to Madison Bruner and Sasha Chen). The credentials to get to the admin interface must be that of one of these two people to add data to the database. Another layer of security is that all computations are done with stored procedures. If the code for the front-end of the application were ever to become available to a user, the actual code behind the operations would still be hidden from them on the database side. Finally, all text boxes are sanitized to prevent SQL injection attacks. Thus, the threat from these kinds of attacks is completely eliminated. The only kind of attack that this system is still vulnerable to is a Distributed Denial of Service (DDoS) attack. This kind of vulnerability was not compensated for and therefore no actions were taken to prevent them because those actions are outside the introductory knowledge of the creators of TLG.

Integrity Constraints

There are many referential integrity constraints that exist within TLG database. These constraints are represented by the arrows in the Relational Schema which is located at the end of the document. The rules for updates and deletes is to cascade so that incomplete or inaccurate data was not used in win rate and other calculations.

Below is a table of the different domain integrity constraints within TLG.

Constraint Name	Constraint Description
Item integer constraint	Integers associated with the Items table can be null and if they are not null, must be greater than or equal to zero.
All other integers constraint	All integers in the database must be greater than or equal to zero. The number of towers and inhibitor had the additional constraints of being less than or equal to four and one respectively.
Item strings constraint	In the Items table, all strings can be null save for the item name. If a string is not null, it must be less than 50 characters.
Positon string constraint	The position string in the Player table must be either 'Tank', 'ADC', 'APC', or 'Support'.

All other strings constraint	All other strings in the database cannot be null and must be less than or equal to fifty characters.

Stored Procedures

Below is a table containing all of stored procedures contained within TLG.

Stored Procedure Name	Stored Procedure Purpose
Champion win rate	Determines the win rate of a particular champion in the database
Death Difference	Calculates the difference in deaths between the two teams of the same match
Exp Difference	Calculates the difference in experience between the two teams of the same match
Final win chance	Uses other stored procedures such as Death Difference, Gold Difference, Item win rate etc. to make the final calculation to tell the user if their team is the projected winner or not
Generate Game	Used by admins to add data to the database; not available to normal users
get avg user level	Gets the average player level in the database
Get matches of champion	Gets all the match id’s of a given champion in the database
get player items	Gets the items of a player in a particular match
get post-game stats	Displays the post-game stats of a particular match including the name, kills, deaths, assists, items, gold, and creep score of each player in that match
Gold Difference	Calculates the difference in gold between the two teams of the same match
Item win rate	Determines the win rate of a particular item in the database
Kill Difference	Calculates the difference in kills between the two teams of the same match

Position win rate	Determines the win rate of a particular position in the database
Admin Authentication	Determines if the user should have access to admin interface

Views

It was decided that no particular views were needed for TLG database. Views seemed unnecessary for this project so none were created on the database.

Indexes

No indexes were made on the database because they did not seem warranted. Almost all of the procedures in the database involve looking at most of the data and used primary keys to locate that data. Thus indexes were not created on TLG database.

Triggers

There are three triggers that exist in TLG database, below is a table describing them.

Trigger Name	Trigger Description
Trigger_on_Player	Ensure the player level and Position within the range
Trigger_on_Summoner	Ensure the summoner level within the range
Trigger_on_Turrets_number	Ensure the number of Turrets and Inhibitors within the range.

Design Analysis

Strengths

- High security measures mean the database is safe from the most common attacks.

- High security measures also mean the database is relatively safe from bad data.

- Constraints for data are in place to make sure anything that is entered is good data.

- Most stored procedures only use primary keys to execute eliminating overhead an index might incur.

- Most procedures, tables, and attributes in the database are clear and concise to anyone who has at least a basic level of experience with LoL.

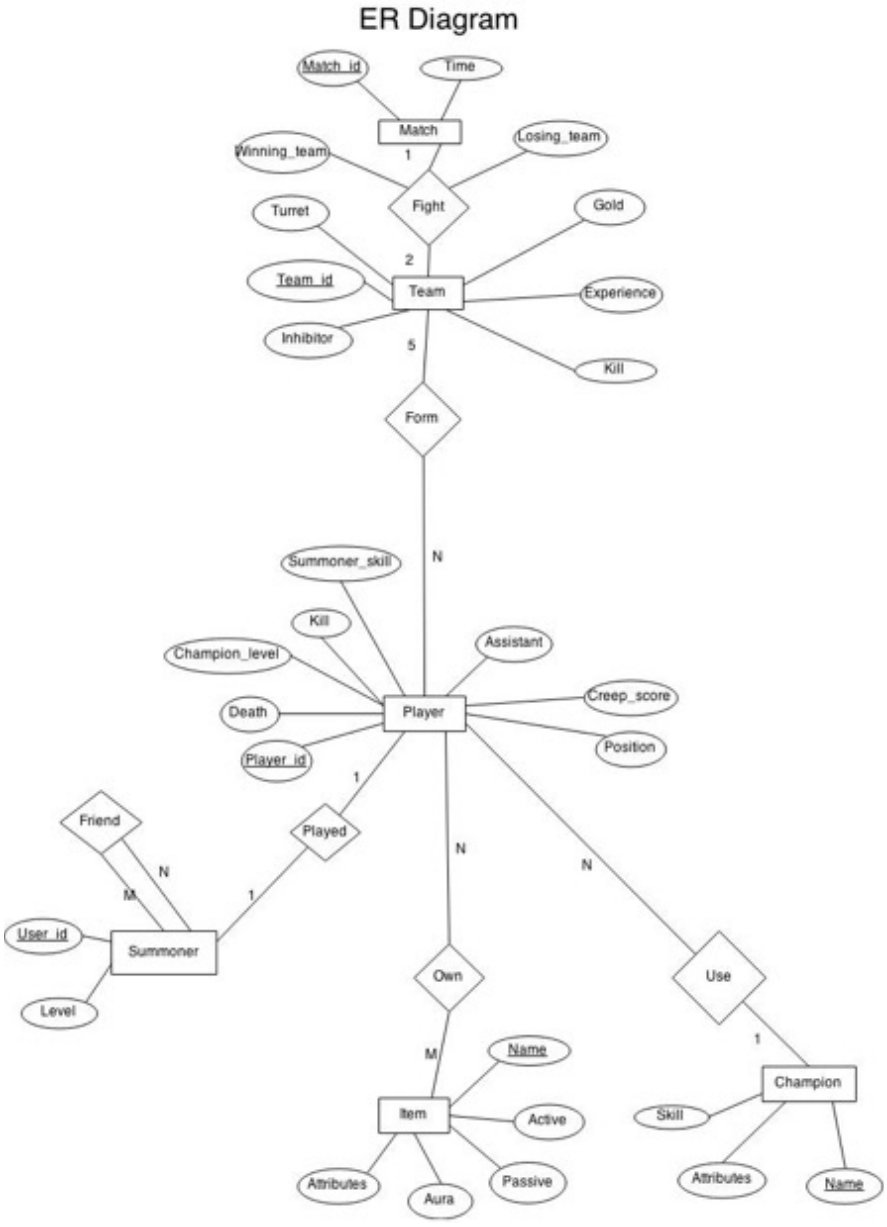
Weaknesses

- Database is still vulnerable to some kinds of attacks. This may be fixed by learning more about DDoS attacks and the different things that can be done to attack a database.

- For someone who knows nothing of LoL, database procedure, table, and attribute names and functionalities may be confusing. A workaround for this problem could have been to include a description of how the game works somewhere within the application; although this doesn't feel like a great workaround, it is the only real way to deal with the problem.

- Several of the stored procedures involve looking at every row for a few different tables. As the amount of data in the database grows, this will cause problems for runtimes. This could be alleviated by having a running total of the different averages that are used in computations. This value could be updated whenever new data is entered into the database and then retrieved quickly when needed for a computation.

Relational Schema



Explanation of Entity Relationship Diagram

A player has an account which is summoner. The instance of a summer in a game is captured by the player entity. The player in a game uses a certain champion and set of items in the match they are involved in. There are five players in every team and a match is formed by two teams fighting each other in a game. The combination of all of this data into a win determination algorithm is the main focus of this database layout.