Team Legend

Final Report

Analysis of the Team Legend Project

Derek Baker
Tyler Gonnsen
2/15/2008

TABLE OF CONTENTS

EXECUTIVE SUMMARY

This final report document begins with a description of the initial problem that created the basis for this project. The specific solution implemented to address that problem, the major challenges of the proposed solution, and the overall design of the database used as part of the solution are discussed following the introduction of the problem. Finally an analysis of the results of implementing the particular solution is performed, citing both the strengths of the particular approach and its weaknesses. An appendix is included at the end of the document containing the final entity relationship diagram and relational schema diagram for the project database. Further document references are provided through a glossary of key terms, a external references list, and an index.

INTRODUCTION

This document is the final report on the Legend project developed by Team Legend, consisting of Derek Baker and Tyler Gonnsen. Its purpose is to take the problem outlined in previous documents and discuss the effectiveness of the solution implemented by Team Legend. In examining the effectiveness of the solution this report will list first-hand feedback from the project team on what they felt were the strengths and weaknesses of their approach as well as revisiting items mentioned in the Security Analysis (1) and Final Problem Statement (2).

PROBLEM DESCRIPTION

Through the success of the movie *I Am Legend* (3) a large fan base was established that is, at the moment, being underutilized by the consumer market. Using the popularity gained from the movie, Team Legend believed that a large opportunity existed to create an online game based in the world found in *I Am Legend* (3). To further the success of the application it was also envisioned that, in addition to being a fun web-based game, the end result would also be an educational experience for users by challenging their minds with puzzles that, if completed successfully, would provide rewards for their in-game characters.

Ultimately, the goal of the development team was to establish a well liked, free version of the game that could then be turned over to a larger company or bring on capital investors that would result in revenue for the original stakeholders of the project. The game would become well liked for its free use, availability over the internet, entertaining *I Am Legend* (3) themed game play, and challenging puzzles.

Below is the list of features of the end system as listed in the final version of Team Legend's problem statement.

Feature #	Feature Name	Feature Explanation
1	Web-based application	The system will be available over the internet
2	Supports major internet browsers and platforms that will support them	Internet Explorer and Mozilla Firefox will each support the software. Any platform that runs them will run the software. Additional platforms such as the Wii may run the software.
3	Varying levels of difficulty	Difficulty varies depending on the puzzles used and the game component becomes increasingly difficult to keep things challenging and entertaining.
4	User experience different with each use	Both the puzzles and the game play vary from day-to-day, creating an experience entirely unique.
5	Upgradeable architecture	The system is designed to support future puzzle types and updates to the game play mechanism.
6	Tie-in with *I Am Legend* (3)	The game takes place in the environment of *I Am Legend* (3)
7	Play affected by puzzle completion	Solving puzzles correctly will be essential to successful game play, while incorrect or uncompleted puzzles will not benefit the user.
8	User experience designed as a game	The main concept of the application revolves around a game so that users find the application fun and enjoyable.

SOLUTION DESCRIPTION

The solution decided upon by Team Legend was to develop the web-application using JavaServer Pages (JSP) (4) and use a Microsoft SQL Server 2005 (5) to maintain all the information related to the game.

FRONT-END DISCUSSION

On the front-side with the JSP graphical-user-interface (GUI), it was decided that the application should be as simple to use as possible. To obtain this simplicity user actions are largely limited to clicking buttons and selecting items from drop-down menus. Free-form input, such as avatar descriptions or names, are available in limited number to the user and have no direct effect on the game itself. Graphics are simple, static images that are predefined within the system. Users have the option of selecting any one of the predefined images to represent their character, however, they are limited only to those images; the user cannot uploaded their own image to be used. Locations and items in the game are also capable of having images associated with them and are also predefined. Users have the option of solving a puzzle each day. Not solving the puzzle will not negatively affect the user's character; however, completing the puzzle will earn them extra items to help them in the game. Only one puzzle is available each day beginning at 12:00 midnight EST and ending at 11:59 PM EST. There are a series of locations the user must move their character through to eventually reach the "goal." As the user progresses to new areas the difficulty of enemies should increase proportionately. Once a user has reached the goal they are awarded with a unique item and remain at the last stage to fight the hardest enemies. As the character progresses across levels their various stats will increase to help them survive the more difficult areas. The application also implements a simple economy system. As the number of a specific item for sale in the game store decreases, its price increases and vice versa. Users can influence the system by making purchases or selling items to the store.

BACK-END DISCUSSION

The GUI runs through the use of the Legend database and the information contained within that. Tables containing information on Users, Avatars, Inventory items, Puzzles, Pictures, and other data interact with the JSP to result in the experience presented to the user. Database access from the application is limited in an effort to provide extra system security. The database currently exists on the Rose-Hulman Dyknow server of the Computer Science and Software Engineering department (6), but will likely be removed after the winter quarter of 2007-2008 ends on February 25, 2008. A more in depth description of the database can be found in the Database Design section.

KEY CHALLENGES

- **Challenge**: Only 2 people available to divide all work among.
  Solution: Work was divided into front-end (GUI development using JSP) and back-end (SQL Server 2005) with Tyler Gonnsen being largely responsible for the front-end and Derek Baker being responsible for the back-end.
  Analysis: This approach worked fairly well. Towards the end of the project it became apparent that there was a greater amount of testing and debugging time required for the front-end work versus the back-end work. In an attempt to compensate for this, all formal documentation beginning with the Security Analysis was created by Derek with Tyler providing the Initial Problem Statement and feedback for all subsequent documents.

- **Challenge**: GUI development was heavily dependent on functionality already existing in the database.
  Solution: Tyler was granted owner privileges to the Legend database during the last two weeks of development and he provided descriptions of necessary stored procedures well in advance when possible.
  Analysis: This worked perfectly. Tyler would provide lists of stored procedures needed days before their actual usage in the program was tested to allow Derek enough time to add them to the database. Due to differences in work schedule, it became apparent that Tyler would need full access to the database to make his own changes if Derek was unavailable.

- **Challenge**: Little experience using SQL Server and JSP.
  Solution: For SQL Server the included 'Help' functionality is generally sufficient, but is also used widely enough that a lot of material exists online to provide aid. JSP assistance was also widely available through the internet.
  Analysis: These resources were sufficient for most questions. Any questions that could not be resolved through online resources were referred to the professor for assistance.

DATABASE DESIGN

For reference of the database design, an Entity Relationship Diagram and Relational Schema are included in Appendix A.

SECURITY MEASURES – STORED PROCEDURES & USER PERMISSIONS

The application accesses the database through a special user account with restricted permissions. All direct interaction between the GUI and database is limited only to executing stored procedures. This provides added security to the system by first off eliminating the need for the application to build its own queries. By removing this there is no chance for a command to be executed that could unintentionally harm the database. Also if someone managed to access the source code for the front-end application, they would be unable to execute malicious queries (i.e. DELETE or DROP) because the account is limited to only EXECUTE. The final measure of security provided is that the possibility of SQL injection attacks is completely eliminated. All data passed to the stored procedures are handled as parameters and are, thus, not made part of the query through concatenation. Aside from injection attacks, the other main security concern that would need to be handled by the database server is the denial of service attack as mentioned in the original Security Analysis (1) document. This is still a viable threat to the system as implementing countermeasures was beyond the time and knowledge associated with this project. The final security concern was handled in the front-end by maintaining special key values that allows only our application to access the JSP files.

INTEGRITY CONSTRAINTS

Various referential integrity constraints exist within the database and are represented as arrows on the Relational Schema diagram included in Appendix A. The general rule for update and delete queries that would violate these constraints is to forbid the transaction, however, a list of exceptions are included in the Security Analysis (1) document.

The following is a list of the domain integrity constraints that exist in the system:

- All integer type values must be greater than or equal to 0 with the exception that primary key values are greater than or equal to 1.
- Avatar.Health must be less than or equal to Avatar.MaxHealth
- User_Puzzle_History.PointsEarned must be less than or equal to Puzzle.PointValue[1]
- User.Email is unique
- Avatar.Status must be one of ‘healthy’, ‘infected’, ‘dead’, or ‘comp’
- Avatar.StartDate must be less than or equal to the result of getdate()
- Picture.Type must be one of ‘AvM’, ‘AvF’, ‘Loc’, ‘Inv’
- Avatar_Relationships.Type must be one of ‘friends’, ‘enemies’, or ‘romantic’
- Inventory.Type must be one of ‘Attack’, ‘Defense’, ‘Health’, or ‘LocAdv’
- Transaction_Log.Timestamp must be less than or equal to the result of getdate()

[1] Enforced via trigger

STORED PROCEDURES

Stored Procedure Name	Procedure Purpose
AvatarReport1	Determines the number of avatars, highest attack/defense/health level, and largest amount of money at each location contained in Avatar_View
CreateNewAvatar	Creates a new avatar tied to the provided UserID parameter
CreateUser	Creates a new user account based on the provided Email and Password parameter combination
DeleteUser	Removes the user account for the provided Email parameter
EconomyReport1	Retrieves the number of items existing for each BasePrice in the Economy_View
EconomyReport2	Retrieves the name and number of that item that are currently being sold by the system store for each item in the Economy_View
EconomyReport3	Displays the name and number of that item owned, total, by all avatars for each item in Economy_View
EconomyReport4	Displays the name, total BaseQty of that item, total QtyAvailable in the system store, and the average BasePrice for that item for each item in Economy_View
EconomyReport5	Retrieves the name, earliest recorded sell date, and latest sell date of the item for each item in Economy_View
EconomyReport6	Retrieves the name, number of people that own it, and total number of avatars in the system for each item in Economy_View
GetAvatarImage	Selects the picture name for the image representing the provided AvatarID parameter
GetAvatarLocation	Selects the picture name, number of days since the avatar started, name, and description for the current location of the provided AvatarID parameter
GetAvatarProfile	Gets the name and description for the given AvatarID parameter
GetAvatarStats	Gets the status, rank, health, money, attack, defense, and maxhealth for the given AvatarID parameter
GetItemsForSale	Selects information used by the system store on every item that can be sold
GetItemStats	Selects all stored information for a given ItemID parameter
GetMaxDefenseItem	Retrieves all information on the strongest defensive item a given avatar owns based on the provided AvatarID parameter
GetPuzzleHistory	Selects all the information about the puzzles which a user has completed
GetPuzzlesList	Retrieves all the information in the puzzle table
GetSellableItemsOwned	Gets the information used by the system store on every sellable item owned by a given avatar
GetUsableItemsOwned	Selects pertinent information on every usable item owned by a given avatar
GetValidAvatarImages	Selects a list of all images that are valid for a given avatar gender
GetValidEnemies	Returns a list of enemy id's that are at the same location as a given avatar
GetWeapons	Retrieves a list of all the weapons owned by a avatar
LoadUser	Returns the avatar id related to a given Email and Password login combination
LookupPassword	Gets the password for a given user Email
MakePurchase	Allows an Avatar to purchase 1 unit of an item
NewPuzzleEntry	Inserts a new row into the User_Puzzle_History table
PuzzleAvailable	Checks to see if the user has completed a puzzle in the past 24 hours
PuzzleReport1	Retrieves the point value and number of puzzles with that point value from Puzzle_View for each point value
PuzzleReport2	Selects the point value, number of people that have passed a puzzle worth that many points, and how many people have failed a puzzle worth that number of points for each point value in Puzzle_View

RecordPuzzleResults	Stores the results for the latest puzzle completed by a user
SellItem	Allows a user to sell 1 unit of an item back to the system
UpdateAvatarImage	Changes the image associated with a given avatar
UpdateAvatarInfo	Updates the name and description for a given avatar
UpdateHealth	Sets the health value for a given avatar
UpdateStatus	Updates the status for a given avatar
UpdateUser	Allows a user to change their email and password combination used to login
UseItem	Allows a user to use an item in their inventory

VIEWS

The Legend database contains 3 views:

- Avatar_View
- Economy_View
- Puzzle_View

They are not accessible through the web-application itself. Instead, they are meant to serve as monitoring tools to execute the various reports from for the benefit of the system administrators. The views are used to reduce the amount of information that needs to be processed for each of the reports.

In general, views are unnecessary for this particular project. Currently, we expect there to exist only two groups of people that interact with the system: developers and players. Players have no direct access to the database and, currently, all developers need equal access to the system. If the project were to be developed further with a larger team, then it may become useful to have various views of data. As it stands now however, there seems to be no reason to filter the data accessible to developers.

INDEXES

Additional indexes defined on the database tables are unnecessary and would only incur unwanted overhead to the system. The majority of the stored procedures execute based on primary key values (e.g. AvatarID, InvID, PiID (Picture ID), ID (Puzzle ID)). There are not enough procedures that consider non-primary key values to justify creating an additional index. Because of this, no additional indexes exist in the Legend database.

TRIGGERS

There is one trigger that exists in the database.

1. **PointsEarnedTrigger**: Exists on the User_Puzzle_History table and activates on INSERT or UPDATE. If the value for inserted.PointsEarned is greater than the value of Puzzle.PointValue for the particular puzzle id, then the transaction is rolled back.

DESIGN ANALYSIS

STRENGTHS

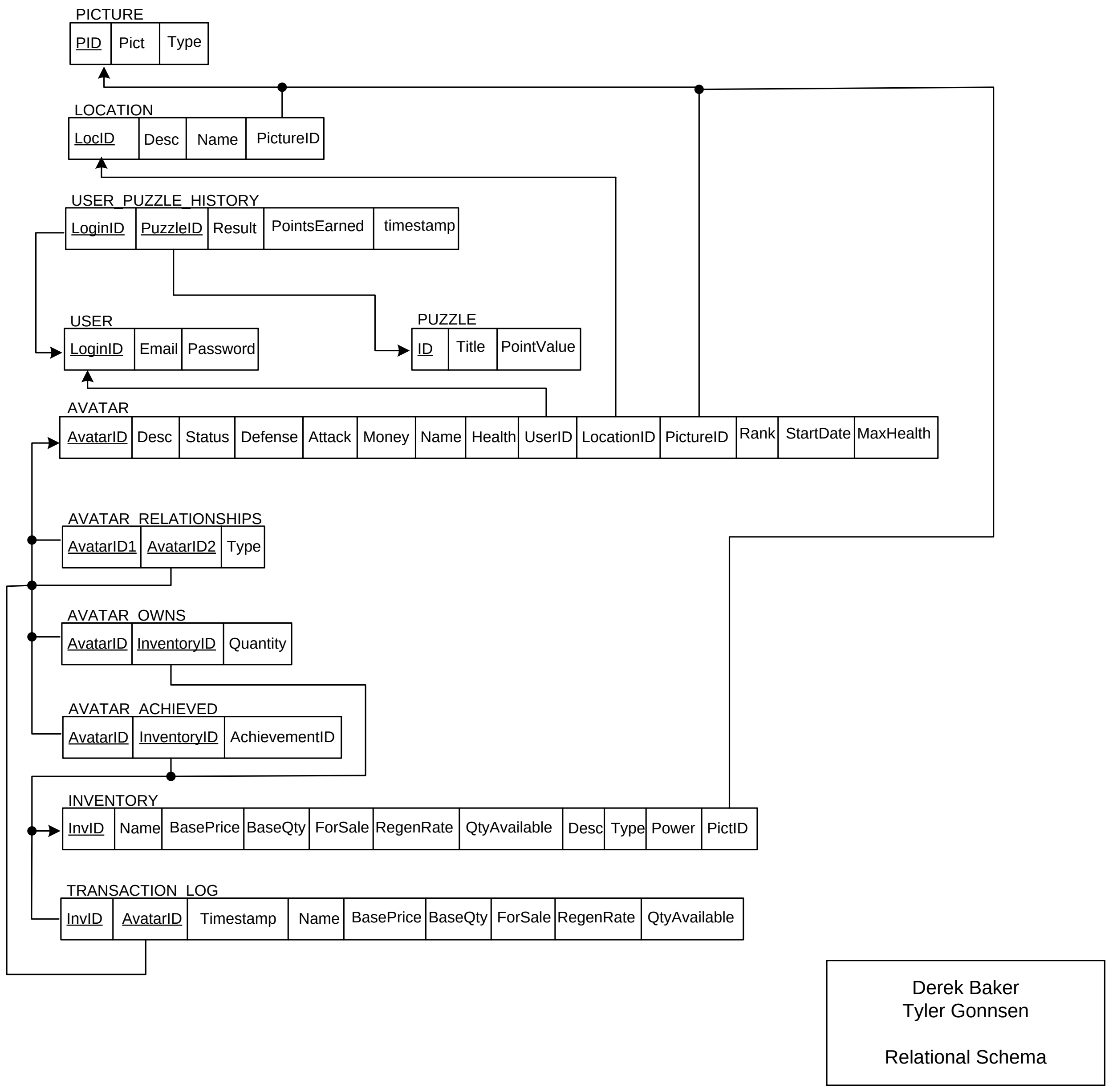
- The system seems to have a high level of security through limiting the actions of the application to only being able to use EXECUTE.
- All system information is equally available to developers and this gives them more information to consider when generating design changes.
- Appropriate domain integrity constraints are in place.
- The default policy of rejecting information changes that would violate referential integrity ensures consistent data in the system.
- Majority of stored procedures examine only primary key fields when executing, thereby eliminating the, potentially, costly overhead resulting from secondary indexes.
- All stored procedures contain a documentation header that describes the general purpose of the procedure.
- Attribute names are relatively clear and descriptive of what they contain. Only a few are ambiguous abbreviations.

WEAKNESSES

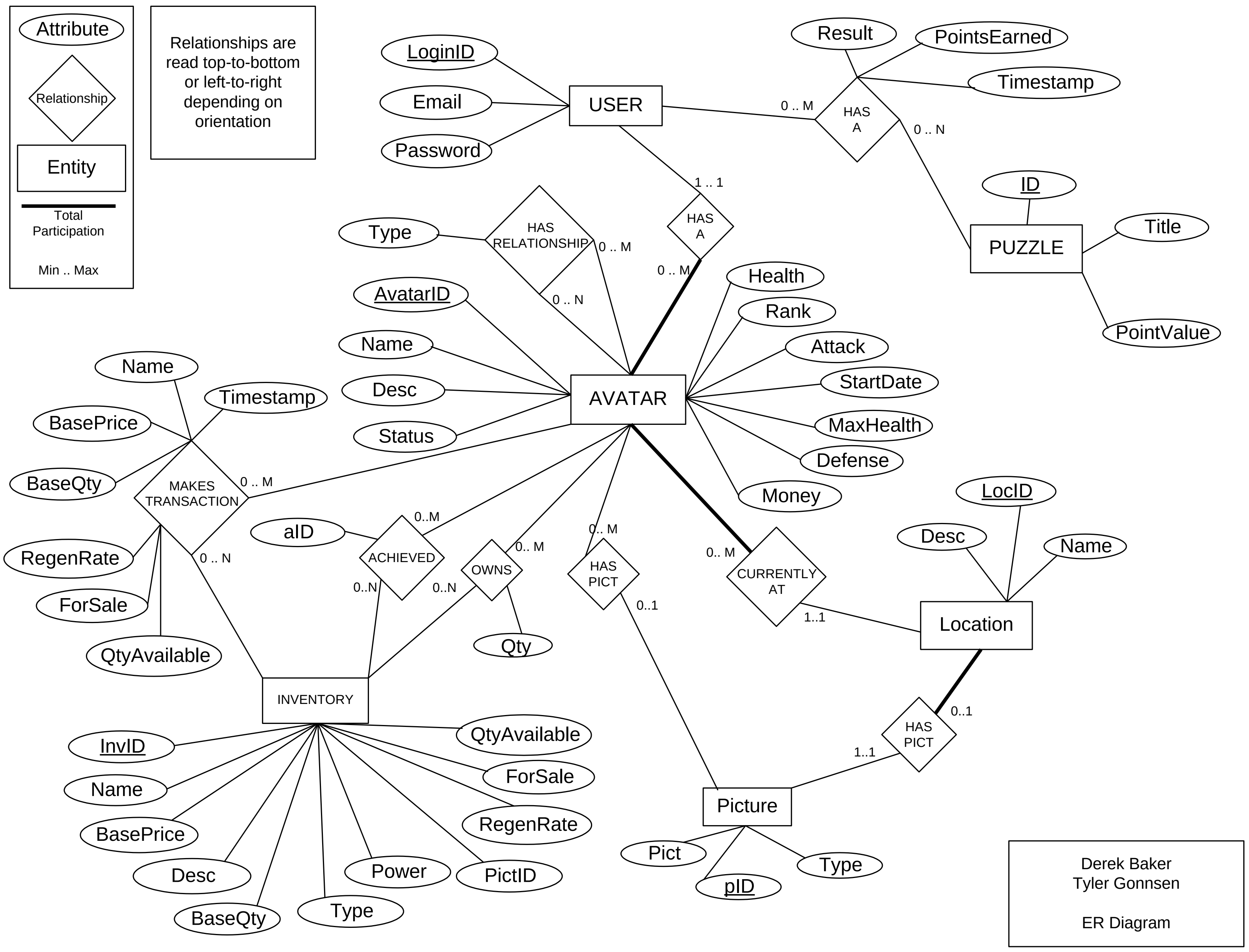
- Extra security also incurs reduced flexibility. Because the application login is restricted to only executing stored procedures, every time a need for additional data is discovered an entirely new procedure must be created to get that piece of information. This could result in time spent idle by developers as they must wait for the procedure to be created in the database before being able to test that piece of functionality.
- The 3 views and the Report stored procedures are used very infrequently and are wasting space that may be needed by the system later on. One way to deal with this is to combine the scripts for the views and procedures into a single SQL query file. These files could then be maintained outside of the database space, brought into SQL Management Studio when need, and executed to view the desired information.
- Passwords are not encrypted within the database and can be viewed by anyone with access to the system. This can be handled by having the JSP application perform an encryption process on the user's password when it is entered. The encrypted password would then be stored in the database and would be illegible to anyone simply looking at the table.

APPENDIX A

RELATIONAL SCHEMA


PICTURE
PID | Pict | Type
LOCATION
LocID | Desc | Name | PictureID
USER_PUZZLE_HISTORY
LoginID | PuzzleID | Result | PointsEarned | timestamp
USER
LoginID | Email | Password
PUZZLE
ID | Title | PointValue
AVATAR
AvatarID | Desc | Status | Defense | Attack | Money | Name | Health | UserID | LocationID | PictureID | Rank | StartDate | MaxHealth
AVATAR_RELATIONSHIPS
AvatarID1 | AvatarID2 | Type
AVATAR_OWNS
AvatarID | InventoryID | Quantity
AVATAR_ACHIEVED
AvatarID | InventoryID | AchievementID
INVENTORY
InvID | Name | BasePrice | BaseQty | ForSale | RegenRate | QtyAvailable | Desc | Type | Power | PictID
TRANSACTION_LOG
InvID | AvatarID | Timestamp | Name | BasePrice | BaseQty | ForSale | RegenRate | QtyAvailable
Derek Baker
Tyler Gonnsen
Relational Schema

ENTITY RELATIONSHIP DIAGRAM



EXPLANATION OF ENTITY RELATIONSHIP DIAGRAM

The system is built from the existence of Users (located towards the top of the diagram). Users can then create an Avatar that exists in the game and is controlled by the User. Puzzles, also located towards the top, are events that can be completed by Users to earn rewards for their Avatars to use. The rest of the system revolves around the use of Avatars and this is shown through its location in the center of the diagram. Every Avatar **must** be at some Location and the Avatar may have a Picture associated with them to represent their appearance. Avatars can also Own items as well as Achieve special items by completing puzzles or reaching certain locations in the game. They can also sell items back to the system, represented by Makes Transaction towards the left of the diagram. The last feature directly involving Avatars is that they can form relationships with other Avatars if they so choose. A final connection to note is that similar to Avatars having pictures, Locations also have a picture associated with them (as depicted in the lower right of the diagram).

GLOSSARY

Avatar: a created personal entity, controlled by a user that exists in a virtual world.

GUI: (graphical user interface) refers to the images displayed on the screen that the user interacts with through the keyboard and mouse.

Index: in the sense of databases, this is a mapping that allows the database system to quickly locate desired data.

Integrity Constraints: limitations imposed on the database to preserve certain relationships among the data and permit only certain values to exist.

JavaServer Pages (JSP): a derivative of the Java language that allows for different types of web documents to be dynamically generated.

Query: a specific instance of SQL code to accomplish a certain task once it has been sent to the database.

SQL: a language used for specifying actions to perform on a database system.

Stored procedures: a stored procedure is a piece of coded functionality that performs a specific task each time it is called and is maintained on the database itself.

Trigger: similar to a stored procedure in that it is a piece of functionality that performs a specific task each time it is executed the main difference is that triggers will execute automatically whenever information in a table is altered in some way.

Views: a limited display of the data that exists in the database tables.

REFERENCES

1. Team Legend documentation. Security and Data Integrity Analysis document. Delivered on: 1/25/2008.
2. Team Legend documentation. Project Problem Statement: Final Version document. Delivered on: 2/8/2008.
3. Lawrence, F., dir. I Am Legend. [Motion Picture]. Warner Bros Pictures: 2007.
4. "JavaServer Pages Technology." http://java.sun.com/products/jsp/. 2/13/2008.
5. Microsoft SQL Server 2005. http://www.microsoft.com/sql/default.mspx. 2/13/2008.
6. "RHIT Computer Science and Software Engineering Department." http://www.cs.rose-hulman.edu/. 2/13/2008.

INDEX