Final Project Analysis

Noire Etoile

5/22/15

Eric Downing

Josh Maurer

Dan Schnipke

TABLE OF CONTENTS

# Executive Summary………………………………………………………….......2

# Introduction………………………………………………………………….…...3

# Problem Description……………………………………………………………..4

# Solution Description…………………………………………….……………….5

## Front End Discussion……………………………………………………..5

## Back End Discussion……………………………………………...………5

# Key Challenges…………………………………………………………………...6

# Database Design……………………………………...………………...………...7

## Security Measures – Stored Procedures & User Permissions……….…7

## Integrity Constraints…………………………………………...…….…...7

## Stored Procedures………………………………………….…….……….8

## Views…………………………….…………………………………..……..9

## Indexes…………..…………………………………………….……..…….9

## Triggers…………………………….………………………….……..........9

# Design Analysis……………………………………………………...………….10

## Strengths……………………………………..………………………..…10

## Weaknesses………………………………………………………………10

# Appendix A……………………………………………………………………..11

## Relational Schema…………………………………………………………….11

## Entity Relationship Diagram………………………………………….......….12

## Class Diagram…………………………………………………………………13

**Executive Summary**

This final report begins with an introduction of the design of the game in order to use the database. We will then talk about our solution with front and back end discussion, followed by key challenges, the design of the database, including everything from stored procedures to triggers, the design analysis of our end product, and finally an appendix that contains our relational schema and ER diagram.

**Introduction**

As a group we decided to build a text based game. Our initial design of the game was to basically be a “Galatic Oregon Trail”. We have everything from planets to star systems included in the database. Throughout this document will further describe the details that encircle our project design and database analysis. In the end we will include a relational schema and entity relationship diagram.

**Problem Description**

We wanted to make a text based game that could be played the way that Oregon Trail was played. The player would start the game as a trader targeting the Galactic Triads good distribution. The player would set out to destroy or remove all illegal goods from the galaxy thus cutting off profit for the triads and winning the game.

Below is a list of features that are present in the game

|  |  |  |
| --- | --- | --- |
| Feature # | Feature Name | Feature Explanation |
| 1 | NPCS | There are non-playable characters in game |
| 2 | Inventory | A player and NPCs can have expansive inventories |
| 3 | Random Items | Items will randomly generate for the player |
| 4 | Planets | There is a multitude of planets |
| 5 | Star Systems | There is a multitude of star systems |
| 6 | Events | Events will randomly happen to player; such as police checking your cargo hold |

**Solution Description**

**FRONT END:**

For the user interface of our game, we chose to use Java outputting to a terminal. We created an engine class that we could use to print to the screen the same way every single time, and an event class to keep consistnacy between events as well. This made use of an interface that was fufilled by a backend Java class that made all the JDBC calls for the project, providing the program will all its interaction to the MySQL database. The user was presented with the same input entry type every single time, which was a line at the bottom of the screen to type into, with the line being entered and the next screen being rendered upon the pressing of the enter key.

**BACK END:**

For the database interface, we used the class called BlackDatabase to handle all of the connections between the front end GUI and the actual database. This class utilized the mysql connector for JDBC in order to call procedures and functions from the database without actually knowing what the commands are, only what they do. It uses prepared calls in which the parameters needed for the functions are set when needed and then the call is executed. Then the ResultSet of the function can be gotten to be parsed by the front end in order to display the desired data.

**KEY CHALLENGES**

* **Challenge**: Using MySQL

**Solution**: Throughout the weeks we spent on this project we came into many issues using MySQL, however, it was just a matter of keeping a close eye on the documentation and being able to read and implement instructions on command.

* **Challenge:** Getting program to run outside of eclipse

**Solution:** In the end we were not able to successfully accomplish this task. We still run it inside of eclipse.

**DATABASE DESIGN**

For reference of the database design, please refer to Appendix A for a relational schema and ER diagram

**Security Measures – Stored Procedures & User Permissions**

Due to the fact that we create a new database for each user the restrictions for the user to user database access is fairly null. However we do protect against SQL injection attacks by formatting the entry of the player name and ship name, as well as, all entries into the command table.

**Integrity Constraints**

Various referential integrity constraints are included in the database to prevent from SQL attacks. We maintain integrity through our stored procedures.

Here is a list of constraints that exist:

* Player Name must be within database to load
* Values must match in trade
* Quantities must be accurate in trades
* Planets must exist
* Star Systems must exist

**STORED PROCEDURES**

|  |  |
| --- | --- |
| **Stored Procedure Name** | **Procedure Purpose** |
| log\_from\_planet | gets the log from the planet |
| Log\_from\_star\_system | Gets the log from the star system |
| Log\_from\_galaxy | Gets log of everything |
| Persons\_inventory | Returns the items that the person is holding |
| Planets\_given\_system | Returns the list of planets from a system name |
| All\_systems\_in\_galaxy | Returns the list of systems in world |
| Travel\_to\_planet | Changes player location to new planet |
| Player\_planet | Gets the player planet |
| Create\_new\_player | Starts a new game with new player |
| Get\_vendors | Gets vendors from given planet |
| Get\_goods\_vendor | Gets the goods from a given vendor |
| Make\_trade | Safely makes trade from player to vendor |
| System\_given\_planet | Returns the system to which a planet belongs |
| Get\_player\_money | Returns the amount of money a player has |
| Get\_used\_weight | Returns the total weight of the user |
| Get\_player\_ship | Gets players ship name |
| Get\_total\_weight | Returns players max weight |
| Get\_all\_goods | Returns a list of all goods in database |
| Player\_exists | Checks to see if a player exists |
| Drop\_player | Deletes player from database |
| Give\_good\_to\_player | Inserts item into players inventory |
| Get\_legality\_good | Checks the legality of a given good |
| Get\_police\_planet | Gets the police level of a planet |
| Get\_danger\_planet | Gets the danger level of a planet |
| Get\_police\_star\_system | Gets the police level of a system |
| Get\_danger\_star\_system | Gets the danger level of a system |
| All\_goods\_illegal | Checks if all illegal goods are gone or owned |

**VIEWS, INDEXES, TRIGGERS**

Due to the attributes of our database we had no need for views, indexes, or triggers.

**DESIGN ANALYSIS**

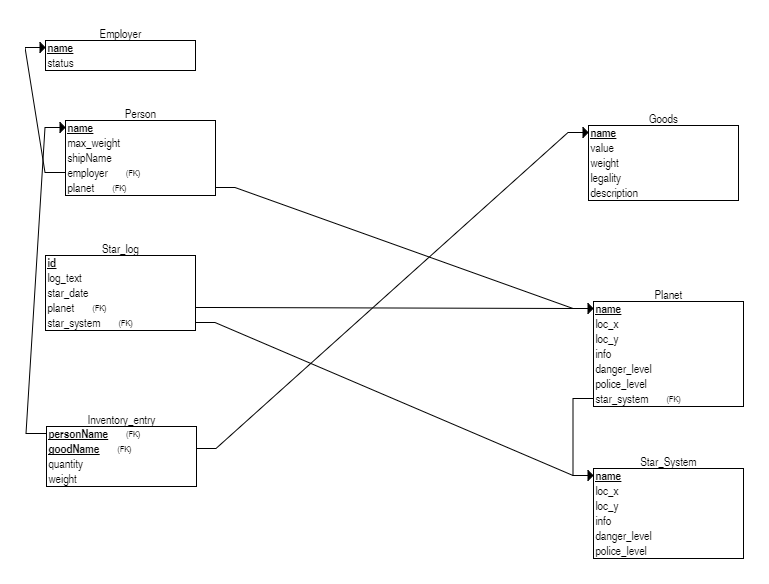
**STRENGTHS**

* Drawing things to the terminal went well, we managed to create a system to always get the terminal filled properly
* Making use of an interface allowed us to improve our workflow as a team
* Using stored procedures allowed us to check our data as we were putting it into the database to prevent duplicates

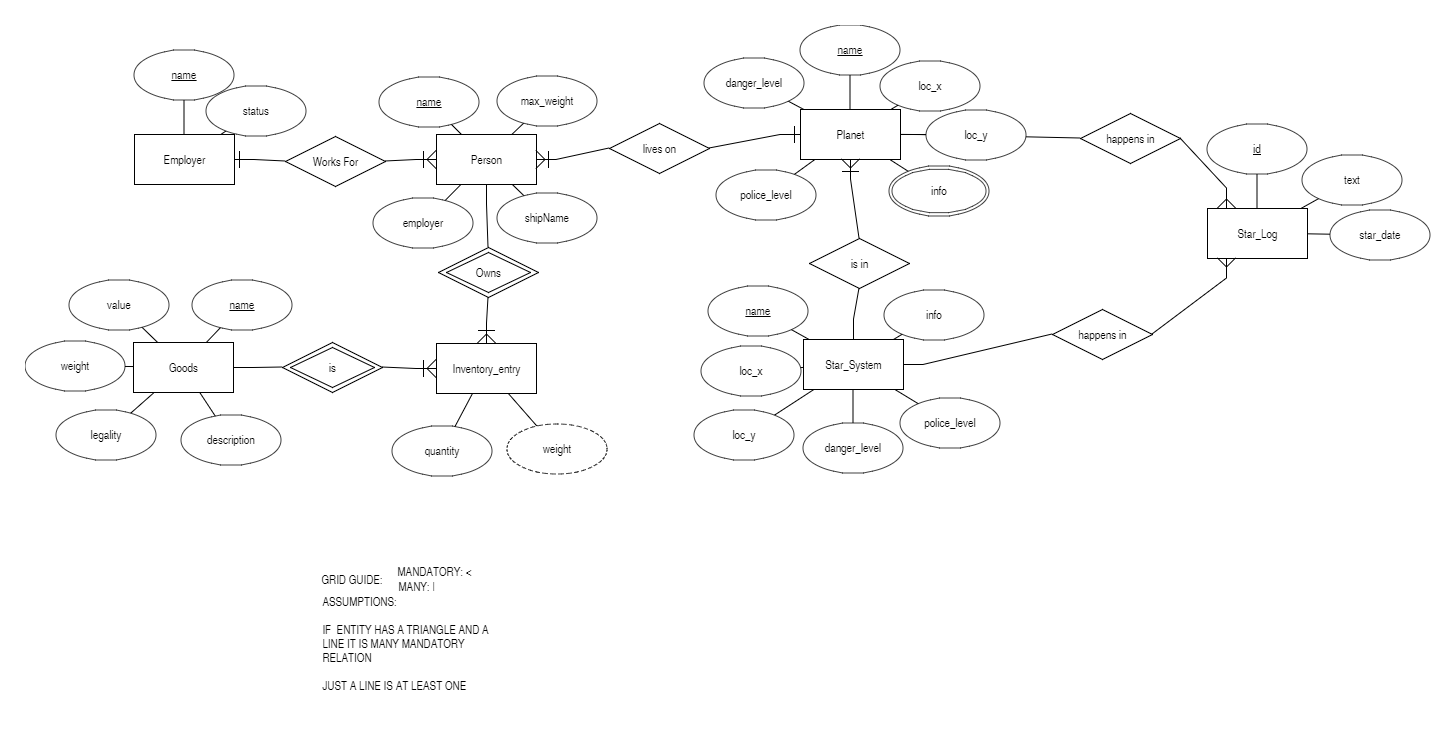
**WEAKNESSES**

* Rolling an external jar into a project so that it can be run outside of eclipse proved extremely difficult, as such, we were never able to get the final version of our project to run in a normal terminal, as was the original plan
* Although we had a streamlined system for rendering each window, we did not have a streamlined system for menus, which quickly became messy and unwieldy, slowing down development speed and leading to many errors and problems

**APPENDIX A**

**RELATIONAL SCHEMA**

**ER DIAGRAM**

****

**CLASS DIAGRAM**

