CONSTRUCTION AND EVALUATION OF PERSISTENT STORAGE FOR AN ONLINE GAME’S STATISTICS

Master Degree of Computing Science

By: Seyed Ali Azimi
cse10asi@cs.umu.se

Supervisor: Patrik Eklund

Umea University
Fall 2013
# Table of Contents

Abstract ........................................................................................................................................... 5  
Introduction ....................................................................................................................................... 6  
Outline of the thesis ........................................................................................................................... 7  
**Chapter 1** ....................................................................................................................................... 8  
Game Industry ..................................................................................................................................... 8  
Taxonomy ........................................................................................................................................... 8  
Game Developing Companies .......................................................................................................... 9  
  a) ROVIO ........................................................................................................................................... 9  
  b) EA .................................................................................................................................................. 9  
Best publishers in the world .............................................................................................................. 10  
Turborilla .......................................................................................................................................... 10  
Planet mule ....................................................................................................................................... 11  
  1. Mule as a strategic game .............................................................................................................. 11  
  2. Game features ............................................................................................................................ 11  
  3. Explanation .................................................................................................................................. 11  
  4. Differences of the new version .................................................................................................... 13  
**Chapter 2** ....................................................................................................................................... 15  
Database Design ............................................................................................................................... 15  
  • Conceptual phase ......................................................................................................................... 15  
  • Logical phase ............................................................................................................................... 15  
  Anomalies and Normalization ........................................................................................................ 16  
  • Physical phase ............................................................................................................................. 17  
Database in Planet Mule (the old version) .................................................................................... 18  
  Database schema ............................................................................................................................ 18  
  Weak points and solutions ............................................................................................................. 21  
  Positive points ............................................................................................................................... 22  
  PHP files for managing queries ...................................................................................................... 23  
**Chapter 3** ....................................................................................................................................... 25  
The logic of grouping queries ......................................................................................................... 25  
  Stored programs ............................................................................................................................. 25  
  Advantages of stored programs .................................................................................................... 26  
  Disadvantages of stored programs ............................................................................................... 27  
  Stored procedures features ............................................................................................................. 28  
  Stored Procedure in MySQL ........................................................................................................... 28  
Concept of Optimization by Stored procedures ............................................................................ 28  
  Time measuring .............................................................................................................................. 29  
  Internal optimization ...................................................................................................................... 30  
Structure of stored procedures ....................................................................................................... 31  
  Showing result set in a stored procedure ...................................................................................... 32  
Stored procedure in different vendors .......................................................................................... 32
# Chapter 4

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>New version of database in Planet Mule-2</td>
<td>33</td>
</tr>
<tr>
<td>Adding / deleting some tables</td>
<td>33</td>
</tr>
<tr>
<td>Modifying attributes</td>
<td>33</td>
</tr>
<tr>
<td>Creating Relations</td>
<td>34</td>
</tr>
<tr>
<td>Merging some tables with one-to-one relationships</td>
<td>34</td>
</tr>
<tr>
<td>Applying Normalization Form</td>
<td>35</td>
</tr>
<tr>
<td>Update Anomalies</td>
<td>35</td>
</tr>
<tr>
<td>Delete Anomalies</td>
<td>36</td>
</tr>
<tr>
<td>Insert Anomalies</td>
<td>36</td>
</tr>
<tr>
<td>Adding indexes</td>
<td>37</td>
</tr>
<tr>
<td>Designing an ER-Diagram</td>
<td>37</td>
</tr>
<tr>
<td>Enhanced Entity Relationship Diagram for Planet Mule-2</td>
<td>37</td>
</tr>
<tr>
<td>Choosing optimized method for using queries</td>
<td>39</td>
</tr>
<tr>
<td>Stored Procedure Code Sample</td>
<td>40</td>
</tr>
<tr>
<td>Diagram of bandwidth</td>
<td>42</td>
</tr>
<tr>
<td>Java interface for calling stored procedures</td>
<td>43</td>
</tr>
<tr>
<td>Conclusion and future work</td>
<td>43</td>
</tr>
<tr>
<td>References</td>
<td>44</td>
</tr>
</tbody>
</table>
# Table of Figures

Figure 1: Phase summary to change race in every round according to condition of the player..............................................................12  
Figure 2: Map of the game for choosing plots..........................................................12  
Figure 3: The place where player should buy mule to produce food, energy, smithore, and crystite in player’s plots .........................................................13  
Figure 4: Auction part of the game for buying or selling player’s products.............13  
Figure 5: Website of the game .............................................................................14  
Figure 6: Storage percentage of tables in the old version......................................19  
Figure 7: Snapshot of tables in the old version......................................................20  
Figure 8: Query statements vs. stored procedure for doing complex operations.......29  
Figure 9: Diagram of execution time in a stored procedure and equivalent query statements in remote vs. local machine.........................................................29  
Figure 10: Relation between Users and Skills tables............................................34  
Figure 11: Relation between Logs and Users_games tables..................................35  
Figure 12: A sample of ER-Diagram .................................................................37  
Figure 13: ER-Diagram of the new database designed.................................38  
Figure 14: Time of calling stored procedure in comparison to query statements’ transfer in the application code.................................................40  
Figure 15: The time spent in a part of project to use queries in comparison to stored procedures (A: Query statements, B: Stored procedures).................................42
Abstract

Online games are gradually getting more popular than traditional video games among players due to their competitive nature, low overhead of installation, and low storage/graphic requirements. Including features such as displaying the latest game logs, top scores and best players makes these games more exciting to the passionate gamers; and this excitement makes them spend more time, try harder to achieve higher scores and be on the top of the score lists. In game industry, the game logs and related scoring data are usually stored in relational databases which grow enormously in size and have to be maintained and remain responsive as a critical component of the online games. Having a well-designed and responsive database structure is a key factor for an online game to remain functioning with the large volumes of data and thousands of concurrent transactions.

In this research, an optimized approach is proposed toward designing a persistence layer and related databases for having an improved storage mechanism for online games. The databases of online games normally have several performance pitfalls such as data redundancy, response time, and high bandwidth usage. To overcome the latter drawbacks, some solutions have been designed and proposed in this study. As a case-study the database of a game project called Planet Mule [5] was re-designed. The new optimized persistence layer will result in a significant decrease in the response time, which eventually leads to the users’ satisfaction and popularity of the game.

This dissertation firstly describes our proposed database design phase which includes conceptual design, logical design including normalization and physical design. Then it continues focusing on the design and optimization of queries with the goal of improving performance. To attain this goal, some standard features of DBMS systems, namely, stored procedures are utilized to optimize query performance and decrease bandwidth usage. Further, proposed stored procedures are optimized internally to achieve a lower response time. Finally, the thesis shows the implementation approach and evaluates the recently designed database layer. The development environment contains a stack of open-source technology or free tools such as Eclipse platform, MySQL workbench, Dream Coder, and MySQL debugger as a debugging and profiling tool over the queries and stored procedures; Programming was done in Java. Our experimental results show that the proposed approach outperforms the old version.
Introduction

Currently, companies that produce online games try to have a website to show their customers the results of the games. Also, customers are more interested in competing with each other in games for achieving better ranking in score tables on the website. So, online games normally involve a database for saving and showing the latest results. In recent years, many companies have tried to have an optimized database to allow more concurrent users to play with faster speed. These companies provide an environment to store and retrieve information without delays or other problems. The database plays the main role in most online games and the collected data are stored during or after finishing the game; therefore, designing an optimized database is a challenge for game developers to produce a real-time game.

Database design is the process of producing a detailed data model of the database and includes some major steps [15]. Database design is part of the overall software development process. First, the requirements should be identified, and then a group of software developers need to analyze the information based on these requirements and investigate how they can be converted to entities. Second, the relations existing between those entities should be distinguished. Third, drawing the ER diagram will make the design easy and understandable for the company or person who orders the game. Finally, after obtaining approval from the client, the database is implemented.

Turborilla Company, owned by Tobias Anderson, is a small company developing video games. One of the most successful products of the company is the Planet Mule project which is a seminal multiplayer video game. Not only it is a strategic game but also it provides an economic simulation. Currently, they are working on a new version of the project called Planet Mule-2. This game can be played on a network of at least two players and is limited to five but if the number of players is less than five, they can be replaced by computer players. One of the main problems they may face is the migration from the old database used for Planet Mule to the new version.

Planet Mule uses an un-optimized database which suffers from lack of any relations between tables, data redundancy, and potential update anomalies. Moreover, the existing queries are not well-written enough to run quickly and they use high bandwidth. The latter issues not only prevent the game from accommodating a potentially large number of future users but also they require the current users to wait during or after playing the game in order to get their result saved or retrieved from the database.

The purpose of this thesis is to develop an optimized database for storing/retrieving the results of the Planet Mule-2. In Planet Mule-2, the application server on which the game should be deployed runs on one server, while the database is located on another server. A large number of users play the game simultaneously and connect
to the game server. Since in the old version, query statements are transmitted directly, by increasing the number of users, a higher bandwidth is required which can be alternatively handled by using stored procedures. The other problem is data redundancy which can be prevented by designing an optimized database in which proper relations are created between the tables, some tables that existed in the old version are merged, and redundant data/attributes are eliminated by normalization.

Outline of the thesis

In Chapter 1 the game industry is described. Also, some of the most famous and the best developing companies in producing games are introduced, two of which are explained in detail. In the second part, the experience of Turborilla Company, owner of the Planet Mule, in producing games is discussed. Finally, the features of Planet Mule are described including a detailed description about how the game is played in order to give information about the existing interactions in the system. This information is used later to understand how and when the data should be stored/restored in/from our proposed database.

In Chapter 2, the required functionalities to design a database are explained. The collected results of the questionnaire from the game developers of Planet Mule are organized in a structured form which later are divided into major entities and their attributes. Then, in the conceptual phase, an ER-diagram describes how these entities relate to each other. In the logical phase, the ER-diagram is the basis for a set of relation schemas which are then used for table definitions. Also, the keys including primary and foreign keys are defined in this phase. Normalization is the last part of logical design and is a technique by which one can modify the relation schema to reduce redundancy. In the physical phase the attributes and the indexes are defined, the keys are created based on DBMS and the SQL clauses are written to create the database. Finally, the integrity constraints and the users’ access rights are implemented. After database design, some positive and negative points of Planet Mule are addressed. Thus removed, created and stable tables are determined for Planet Mule-2. Also, some PHP programming codes used in Planet Mule are described to apply the required changes in the new version of the game.

In Chapter 3 some features of stored programs including their advantages and disadvantages are indicated. Next, the existing queries are optimized by internal optimization and comparing its elapsed time to equivalent stored procedure in order to decrease the volume of data transferred on the network. In a stored procedure, queries are compiled on a database server and the program can work on these data from within the server, rather than having to transfer the data across the network. Hence, the stored procedure can decrease the traffic of the network.

In Chapter 4 the proposed database for Planet Mule-2 is described, and the new changes in database and application code are shown. In addition, the normalization forms are implemented step by step on existing data in the system to avoid data redundancy. Apart from these, our best result which is an optimized stored procedure code sample is described.
Chapter 1

Game Industry

Game is an interactive entertainment, and the related industry is an economic business that consists of marketing and development of video or computer games to millions of people in the world. There are over 11 countries with the income of over 1 billion dollars which can attract a great deal of attention. First of all, most young people like to spend their leisure time playing games and there are also some people who play games to earn money. Second, there are lots of companies in the world which distribute those games globally.

Taxonomy

There are different ideas about game taxonomy. These ideas are classified depending on or independent of their content or setting and are not based on other works of fiction such as books or movies. Games are globally classified according to three major categories:

- Game physical’s requirements (requires equipment, size and nature of playing field, and so forth in addition to players)
- The structure of the game (requires number of players, grouping of players strategies, and so forth)
- Games personal requirements (requires motor skills, fitness level, numeracy, social skills of the players and so forth)

There are some other categories for classifying games which are mentioned below.

a) According to number of players:

- 2 players board games such as chess
- Multiplayer card games such as poker
- Marketed family games such as monopoly
- One player such as puzzles
- Zero player that uses artificial intelligence and is determined by its initial state, requiring no further input from humans such as Conway's Game of Life

b) According to game structure [14]:

- Simulations
- Educational or informative
- Sports
- Sensorimotor (e.g. action games, arcade games, fighting and shoot-them-up games, and driving and racing simulators)
- Action-adventure
- Adventure
- Role-playing
- Other vehicular simulators (not covered by driving and racing)
- Strategy games

Game Developing Companies

There are a lot of games companies in the world and some of them are popular among the people who play games. It’s a good idea to mention two of them here.

a) ROVIO

ROVIO [12] in Finland is the owner of the Angry Birds, one of the most popular games in the world. This company sells a lot of games yearly and there are a large number of fans for the games made by this company especially Angry Birds. One of the notable things about this company is having nearly 10 million likes on Facebook. The company started to work in 2003 as a mobile game development center and won a lot of awards. It also has extended its activities to a variety of new business areas such as broadcast media. For example there are a variety of versions based on operating system for Angry Birds and some of them are as follows:

- Android
- Mac
- PC
- Palm
- Nook Color
- Google+
- iOS

b) EA

Electronic Arts Montreal is the producer of battlefield [13], FIFA 2010, etc. and it is located in California. Making a lot of games in different platforms and buying some other companies are its bold specifications. For example, Battlefield originally is the name of a company from Sweden and it was bought by EA in recently. Nowadays there are a lot of platforms in the world where games are made. EA Company has expanded its activities in all areas some of which are worth mentioning:

- PC Games
- Xbox 360
- PS3
- iPhone
- iPad
- Mobile
- Wii
Best publishers in the world

According to Metacritic website that keeps the score of entertainment, the best publisher in getting score of the games in 2010 are respectively [10]:

1. T2 (take two interactive) 2. Nintendo
3. Capcom 4. Microsoft
5. EA (Electronic Arts) 6. Activision Blizzard
7. Sony 8. Square Enix
11. THQ 12. Ubisoft
13. Konami

In the year of 2010 at the Montreal, three biggest companies based on the number of employees in video game are listed in below [9]:

1. UbiSoft Montreal is a company with 1800 employees that has the first position of this ranking. Every year, this company publishes a lot of games which have some of the best rankings among all games.
2. Electronic Arts Montreal is in the second place with 500 employees.
3. A2M is in the third place with 450 employees.

Turborilla

Turborilla is a small company which has just three developers who have produced a few funny and innovative games. Some of these games can be played on cell phone, iPod and computer on the net or locally. Apart from planet M.U.L.E they have some other products such as Mad Skills Motocross game. Tobias Anderson started the company in the summer of 2007 and worked on Mad Skills Motocross for Mac/Win/Linux as well as making a few small games [11]. He studied CS in Luleå and did CS consultant work not related to games and has previously worked to create games as a hobby in his spare time.

The second person that has worked for this company is Peter Sundqvist. He is the creator of Planet Mule and Planet Mule-2. He was employed one year later to work on Planet Mule. That game was ordered by a foreign customer so it was a consultant job, though they very much affected game design as well. It was completed in about a year and then regularly updated for some months after that. After Planet Mule he started working on Planet Mule-2 for the same customer. He studied Computer Science in Umeå and has made several free games on his spare time.

Tobias and Peter also worked on Mad Skills Motocross for the iPhone and released it at the start of 2011. They did this together with an American motocross magazine called Racer X. Michael was hired at the end of the development phase to make updates to MSM iPhone. Michael studied game
development at PowerHouse in Kramfors and did his internship at Resolution Interactive in Umeå, making games for iPhone. Since Mad Skills Motocross sold well on the iPhone, they moved on and are concentrating on their own games. Currently they are working on Mad Skills BMX and have plans to start on other games as well. Planet Mule-2 is still not finished but its developers do not have time to complete it due to all other projects. They are therefore looking to hire someone to complete that game and start working on their next own title.

**Planet mule**

1. **Mule as a strategic game**

A strategic game is a game in which players should have decision making skills. So, ability of the players is important in the result of the game [71][22]. There are some special auctions in this type of game and the players can implement some auction based task such as determining price and advertising budgets. In the end of each step the results diagram will be shown.

Fictional planet Irata is set on a game which is called M.U.L.E. This game is a strategic game comprising educational aspects and its providers have a lot of experience in terms of strategic games [8]. Planet Mule-2 is the current version of this game.

2. **Game features**

Basically, users should register in the forum of the website and download the game free and install it on PC. A chat room was provided for the game and the competitors can send messages for invitation on its chat room before starting the game. There is also chatting possibility during the game. According to number of players and their assets, time of each round is different. Planet Mule can be played on the network between at least two players and is limited to five in the new version but if the number of players is less than five, they can be replaced by computer players. There is also a spectator role for each player who just wants to watch the game.

3. **Explanation**

There are twelve rounds in the game and having good knowledge about auctions and managing own money is an important skill in every round. During and between these rounds the players should manage their money by buying or selling their products. The products existed in the game consist of Food, Energy, Smithore and Crystite. Two positive points are creating more time in the next round of each game by earning more food, and producing more products in factories by having more energy. Also, Smithore is needed for producing a plot when it finishes and Crystite is needed for selling and getting more score. According to players’ scores and resources, the players can decide to sell or buy their products which are what they have to do for going to the next round. Each round of the game consists of 5
separate phases:

- Land Grant/Auctions
- Development
- Production
- Auction
- Summary

Every phase has an important role to increase score. Some snapshots of the game are given in some phases:

Figure 1: Phase summary to change race in every round according to condition of the player.

Figure 2: Map of the game for choosing plots
Figure 3: The place where player should buy mule to produce food, energy, smithore, and crystite in player’s plots

Figure 4: Auction part of the game for buying or selling player’s products

4. Differences of the new version

Planet M.U.L.E is a free reproduction of original M.U.L.E game. The original designers are Dan Bunten & Ozark Softscape who made it in 1983, but in 2009, Planet Mule was made by Turborilla. Nowadays the new version of this game (Planet Mule-2) is going to be published worldwide and everyone will be able to download it from its website, connect to each other and play it. A snapshot of the
The game’s website is shown below [5].

Figure 5: Website of the game

There are some important differences between the features of the old version and the new version of the game. Some of them force some changes in the database.

- The bold difference is the ability of undoing time of the game to an earlier time. In this case, the players can make decisions more precisely because this capability enhances the game flexibility.

- There is a ranking system in the website of the new version that is not similar to the old version and is related to one of the best calculation procedures in the world. The company has not decided to choose any special algorithm.

- There are some races which the players can choose from and changing races in each round is one of the special points of this version. The player has the ability of changing their own race in every round according to race’s capabilities which are mentioned in the game. Every player can select what they need for the next level and choose the correct type of race according to the remaining colonies for the next round.

- The graphic, the shape of races and the size of the map are also changed and the game is more realistic than before.

- Details of the game aside, in the old version every game is hosted by one of the players while the new version of Planet Mule has two servers. On the one side, there is a server for hosting the game, running the application server and manipulating the data. On the other side, there is another server for maintaining those data in a database.
Chapter 2

Database Design

A database must serve the functions of an organization. If the underlying database is not well designed, the application performance will never be satisfactory. If the database is designed properly, the programming of the application is easy to do and does not require extra work. A well designed database offers many ways to fetch data and should be flexible, work well and give the right data and adapt itself to future growth.

Specified requirements can be satisfied utilizing the architecture and data in a system design. Database design is involved in the system development lifecycle and is part of the system design. Before starting database design, data planning is done to allow the stages of the database system development lifecycle to be realized as efficiently as possible. Next, system definition for determining the scope and boundaries of the database application are described. Then, the requirements are collected and analyzed by getting information from the customer. Finally, the database is designed step by step in detail.

Database design is the process of producing a detailed data model of a database and is divided into three phases: Conceptual, Logical and Physical.

- **Conceptual phase:** In this phase, the model of data which is based on the requirements specification of the system is constructed to be used independent of all physical considerations. So a conceptual schema is created based on the collection of analyzed requirements. The result of this schema is an Entity-Relationship (ER) Diagram or UML class diagram that is the high level data model of the specific application area. ER diagram shows what attribute(s) each entity has and how different entities are related to each other and determines the cardinality of each relationship. The cardinality shows how many entities of one side of the relationship belong to how many entities of the other side of the relationship (One-to-one, one-to- many, many-to-one, and many-to-many). In other words, the cardinality measures uniqueness of data in a column of a table and every value with a lower cardinality duplicates more in a column. ER diagram can be designed in a reasonable form by some existing software on the computer such as visual paradigm. Also the basic data model operations can be used to specify the high-level user operations identified during the functional analysis. There are several notations to draw an ER diagram.

- **Logical phase:** In the logical phase or the data model mapping phase, the model of the data to be used is based on a specific data model such as relational data model, but it is constructed independent of a particular database management system. The ER diagram is transferred to relation schema which is the basis for table definitions. Also, the primary keys and foreign keys are defined in this phase. It is important to consider some notions to choose keys efficiently. There
are three important guidelines for selecting a suitable key:

- **Stability**: they should not change frequently.

- **Familiarity**: the people who use keys should know them and what they mean.

- **Simplicity**: the keys chosen should be as simple as possible and not too long.

Apart from these guidelines, a surrogate key is a type of key that uniquely identifies a row for all uses such as join operations and is preferable to a natural key. If a natural key is used and the application inserts one record with incorrect key to the database, it is necessary to change all tables that have its reference in order to update the row. The second problem is the difficulty of finding a unique natural key as they are often of type string, while finding the numeric field is more efficient than a string one [18] [19] [20].

In addition to choosing keys, the logical phase contains applying normalization forms on existing entities which have redundant data.

### Anomalies and Normalization

Anomalies are inconvenient or error-prone situations arising when the tables are processed. There are three types of anomalies:

- Update Anomalies
- Delete Anomalies
- Insert Anomalies

Normalization is a data analysis technique to design a database system. It allows the database designer to understand the current data structures in an organization. Furthermore, it aids any future changes and enhancements to the system. In other words, normalization is a technique for producing relational schema with the following properties:

- No Information Redundancy
- No Update Anomalies

The end result of normalization is a set of entities, which removes redundancy (i.e. duplication of data) and avoids the anomalies which will be discussed next.

### Normalization Stages

Normalization stages involve applying a series of tests on a relation to determine whether it satisfies the requirements of a given normal form or not.
• When a test fails, the relation is decomposed into simpler relations that individually meet the normalization tests.
• The higher the normal form, the less vulnerable to update anomalies the relations become.
• Three Normal forms: 1NF, 2NF and 3NF were initially proposed by Codd.
• All these normal forms are based on the functional dependencies among the attributes of a relation.

Normalization follows a staged process that obeys a set of rules. The steps of normalization are:

**Step 1:** Selecting the data source and convert into an unnormalized table (UNF)

**Step 2:** Transforming the not normalized data into first normal form (1NF)

**Step 3:** Transforming data in first normal form (1NF) into second normal form (2NF)

**Step 4:** Transforming data in second normal form (2NF) into third normal form (3NF)

Occasionally, the data may still be subject to anomalies in third normal form. In this case, further transformations have to be performed.

• Transforming third normal form to Boyce-Codd normal form (BCNF)
• Transforming Boyce-Codd normal form to fourth normal form (4NF)
• Transforming fourth normal form to fifth normal form (5NF)

**Physical phase:** In this phase, the **DataBase Management System** (DBMS) is chosen and the implementation description of the database is created. Different DBMSs support different data types. DBMS as a part of database design is a software system designed to allow series of operations for creating, maintaining, querying and administration of databases. The general purposes of these operations are [1]:

• **Defining** a database consisting of data types, structures, constraints, etc.
• **Constructing** the database involving storing the data
• **Manipulating** the database such as making query to retrieve specific data.
• **Sharing** the database letting users or programs access data.
• **Protecting** or **maintaining** data

The base relations, integrity constraints, security, indexes, the users’ access rights are defined using the SQL language in physical phase. Integrity constraints include the domain integrity, the entity integrity, the referential integrity and the foreign key integrity.
• **Domain integrity:** The Domain contains the values that are valid for the attributes of tables such as data type, length, uniqueness, nullity, range, etc. These definitions ensure that a specific attribute will have a right and proper value in the database.

• **Entity integrity:** This integrity states that no primary key can be null and the primary key must have a right value to identify individual rows in a table.

• **Referential integrity:** It includes some rules which maintain consistency among rows between two tables. For example, a record cannot be deleted from a primary table if matching records exist in a related table.

• **Foreign key integrity:** It consists of cascade update related fields and cascade delete related rows. These both are rules over the referential integrity rule.

The structural database should be implemented after completing these phases and the customer’s approval in order to ensure that the design is satisfactory and free of problem. The term database design is not just used for designing the structure of data storage; it is used to apply overall process of designing including the queries used in the database application. The query statements that are based on existing functional requirements in the application should be declared and managed efficiently.

The structure of data storage leads to Data Definition Language (DDL). A data definition or description language is something like a programming language to define database schemas. Making queries leads to Data Manipulation Language (DML). DML have their functional capability organized by the initial word in a statement, which is almost always a verb for Inserting, deleting, updating and selecting data. DML and DDL are the subsets of Structured Query Language (SQL) [1]. SQL statement contains DDLs such as CREATE, ALTER, DROP and DMLs such as INSERT, UPDATE, DELETE, and SELECT.

**Database in Planet Mule (the old version)**

The old version of *Planet Mule* has three databases to maintain three organized collection of data. One of them is *Forum* database which maintains the data existed in the forum of the website. The website has a forum for the people who play this game with other players and the producer of the game to communicate. The other one is *WordPress* database that consists of web pages required data on the game’s website. The website is made by a blog tool called WordPress [16] and can be edited online by logging in to the Word Press admin area. In short, the WordPress database stores data for the options selected in WordPress. The last one is *Mule* database which is used for storing the results of past games. Since the results of the other games and updating positions of players motivate them to start a new game and encourage them to invite the others, the Mule database is more important than the other two databases to be efficient. So, *Turborilla* has requested a new and optimized database design for *Planet Mule*-2. Database schema in the old version will be described to find out the positive and negative points.
First of all, a chart of tables’ size in the old version of Planet Mule is shown below:

![Figure 6: Storage percentage of tables in the old version](image)

According to the chart, there are some big tables which make optimizing the database an important task. Round_Results table is the biggest one which saves the results of players at the end of each round. Users_Games table which maintains the information of every user in each game\(^1\) is the other one. Also, Logs table which maintains the path of log file for every user in a game is the third one. Optimizing data manipulation on these big tables and keep their redundancy at the minimal level improves the database efficiently.

Designing the ER-diagram for Planet Mule can make tables and corresponding relations understandable to create proper database for the new version. In figure 7, an ER-diagram for the old version of game’s database is shown. As shown in fig. 7 there are neither any relationships between tables nor any integrity constraints on them, but instead some constraints exist which are just handled by the application. In addition, there are some extra tables which can be removed from the database.

---

\(^1\) It maintains all players information even AI players
Figure 7: Snapshot of tables in the old version

- Removed Tables

There are some tables that will not be used in the new version and should be deleted from the database. These tables are listed below:

- **EVENT_TIMES** table that stores the events which happened in the game such as changing the time of each game and starting new week to refresh the scores table.

- **LOBBY_CHATS** table which is for saving the chatting log between players. Lobby chat will not be used in the new version.

- **PLANET_MAPS** and **PLANET_MAPS_COORDS** tables that are for a map which is uploaded into the game website. They will not be used for the new website.

- **HOST_SCORES** table which stores scores of hosted players. Since, there is a server for hosting all games in the new version of the game, this table will be deleted.
• **BADGES** table that is for saving the badges which are given to players during the games. There will be not any badges for the players in the new version.

Apart from the above tables, other tables will remain in the new version of game’s database and are divided into two groups:

• **Merged tables**

  **SKILLS, CHATS** and **LOGS** tables are merged with some other tables to improve fetching data, optimizing redundancy and some other reasons that are explained later.

  • **SKILLS** table is for storing the data which is required to calculate the rank of every player by a true ranking system algorithm and is earned according to the player’s skill in each game.
  
  • **CHATS** table stores the content of chat between players in every game.
  
  • **LOGS** table contains the event log of every player in each game.

• **Stable tables**

Other remaining tables are mainly stable and just have some changes.

  • **USERS** table is for storing the users that are registered in the website whether they play any game or not.
  
  • **GAMES** table is for the games which were started whether they are finished or not.
  
  • **USERS_GAMES** table contains the information of every user in each game consisting of scores, etc.
  
  • **ROUND_RESULTS** table is for storing the result of players at the end of each round.
  
  • **USERS_STATISTICS** table is for storing the points of each player collected during the games.
  
  • **RANKS** table contains the information of ranking scores for each player according to the number of winning games.
  
  • **TOP_USERS** table contains the scores of players during each week to see who the best player is in current week.

**Weak points and solutions**

There are some weak points for the old database which are mentioned below:

• Since there are not any relationships between tables in the old database, wrong operations on data are just prevented by the application layer which is not very robust. Programming errors, ultimately lead to the insertion of false records and finally database will be corrupted. Some integrity constraints can be determined for the relationships made between parent and child tables to prevent wrong operations. They are explained below:
• Not Null constraints on the foreign key
• Unique constraints on the foreign key
• Not Null and Unique constraints on the foreign key
• Other constraints such as Referential and foreign key Constraints

• There are some one-to-one relationships between tables which can be merged by adding some columns of one table to another one. These tables are:

  • Skills table columns for every row in Users
  • Chats table columns for every row in Games
  • Logs table columns for every row in Users_Games

• There are two kinds of ranking system whose computing can affect the time of the game:

  • The first one is a very simple ranking system and is based on the number of games which have been won by the players and are shown in statistics.
  • The second one is computed based on the highest skill in the Skills table.

Currently, the former is not a standard ranking and is not often used. There is a true skill for ranking system which is used commonly in most games. Update of the Ranks table after finishing every game can be a bottleneck because running True Skill algorithm takes long time (a bottleneck is a part of the code that takes the longest time to run). Similarly, updating each player's rankings and sorting the position of players in Ranks table takes much time. For the new version of the game, the company is going to have two ranking systems like the old one. They have not decided yet how to implement the new ranking system based on the True Skill. Maybe they will change the way because of some probable bottlenecks.

Positive points

The old version of Planet Mule database has some specifications which are admired. Some of them are explained below:

• Choosing composition of primary keys instead of making one surrogate key is a positive point in the Users_Games table [17]. The unique id of this table is composed of Game_id column which is referenced by the primary key of Games table and the user number column. In this case, using this composite key makes retrieving Game_Id less costly than fetching data by joining the key of Games tables. There is another composite key in the Round table that is similar to Users_Games and is extended with the composition of round number column.

• Choosing a primary key is affected by how the corresponding tables are used. For example, the primary keys of User_Statistics and Ranks tables consist of
user_id in composition to the Mode Column which is of type string. Since the value of Mode column is just a string and is fetched many times, it is better to use a referenced id instead of string value for this column. Also, this column has a low number of values and it is preferred to reference the id of Enum data type instead of a new table. Proposed Enum column can be merged further with User_Id as a composite key.

Therefore, having a single indexed composite primary key that constrains our data uniquely is less costly than an extra column, extra index and an extra unique constraint on a column in a new table.

**PHP files for managing queries**

Some PHP files were written in the project to manage CRUD operations in the old version of game’s database. These files were called by an http action in a post method or a session written in the application codes. The properties sent to PHP files were taken and used in the queries and a response was returned to the java application. Let’s describe some important definitions used for PHP programming to apply the required changes in the new version of the game:

- Defining a property in PHP is done by putting a $ at the beginning of a name. It is defined such as $user_id.
- To get property sent from the session or post method in application $_SESSION['name of property'] or $_POST['name of property'] can be used.
- `mysqlquery($sql)` function is used to compile the query and get a result set.
- For every row `mysql_fetch_object($name of variable returned by database)` is used to fetch the details of returned data.
- For the errors like SQL exception `trigger_error('related message')` is raised.
- To show every attribute, there is an arrow that is directed towards that attribute e.g. `$user->user_id`.
- Similar to any other languages there are some statements such as if, while, for loop, for handling queries.
- Calling another PHP file is possible by the command of `require_once 'name of the PHP file'` [3] [4].

One of the largest PHP files in this project is shown below:

```php
<?php
  // Sent by the game host when the game has ended normally
  require_once 'inc/begin.php';
  $game_id = $_SESSION['game_id'];
  $user_id = $_SESSION['user_id'];
  $time = getDateTime();

  // Get game mode and rounds
  $sql = "SELECT mode FROM games WHERE id={$game_id} AND host_user_id={$user_id}";
  $games = mysql_query($sql) or trigger_error(mysql_error());
  $game = mysql_fetch_object($games) or trigger_error("Game with id \{$game_id\} doesn't exist");
```

![This is the first query](image-url)
$mode = $game->mode;

// Find all users in this game
$sql = "SELECT user_id, disconnected, player_index FROM users_games WHERE game_id={$game_id}";
$users = mysql_query($sql) or trigger_error (mysql_error());

// A game is legal if it ends with two human players
$numHumans = 0;

$user_ids = array();
while ($user = mysql_fetch_object($users)) {
    if ($user->user_id != 0 &&
        $user->disconnected != true &&
        $user->player_index >= 0)
        $numHumans++;
    array_push($user_ids, $user->user_id);
}

$legal = ($numHumans > 1) ? 1 : 0;

// Set game as finished/failed and legal or not
$sql = "UPDATE games SET status='finished', legal={$legal}, failed={$failed} WHERE id={$game_id} AND host_user_id={$user_id}";
mysql_query($sql) or trigger_error (mysql_error());

// Update statistics for all players in the game
require_once 'update_user_statistics.php';
updateUserStatistics($user_ids, $mode);

// Update weekly the top score list
require_once 'update_top_users.php';
updateTopUsers();
respond(“success”, “true”);

// Notify the games changed event
require_once 'events.php'; (Omitted)
muleEvent('games_changed');

// Update ranks
require_once 'update_ranks.php';
updateRanks(false);

// Update hosting score
require_once 'update_host_score.php';
updateHostScore($user_id);

// Update skills
require_once 'update_skill.php';
updateSkill($game_id);

// Update the ranking based on skill
require_once 'update_skill_ranks.php'; (Omitted)
updateSkillRanks();
Chapter 3

The logic of grouping queries

One of the biggest challenges for database management systems is the high cost of executing complex queries. The companies which distribute DBMSs try to find a solution for reducing the time of executing these queries and the traffic of network. Logic of grouping queries was proposed by these companies to solve the problems of efficiency and performance, because grouping queries in a single place and compiling them on a database server is quicker than transferring data across the network many times. One approach for implementing the logic of grouping queries is to create stored programs including query statements.

Stored programs

A stored program is a series of instructions associated with a name which are stored and compiled within a database server. Stored programs are executed within the memory address of a database server process and affect the performance of systems. There are three types of stored programs [2]:

- **Stored functions**

  Stored functions can be evaluated for each row in a query as a block structured language that usually don't support OUT parameters and they have a return value. These limitations make functions useful for some special tasks. Also they are compiled every time they are requested.

- **Triggers**

  Triggers are fired in response to an activity such as insert, update or delete and are used for data validation. A trigger lets programmer write a set of query statements which can be used by multiple applications and avoid redundant code when multiple programs need to perform the same database operations. Also, it can create an audit trail of activity and implement a business rule, etc.

- **Stored procedures**

  A stored procedure is explained as a block structured language that has some commands for manipulating variables, managing and doing conditions, performing and controlling iteration, handling exceptions, etc. To put it simply, a stored procedure is a group of Transact-SQL that are compiled to a single place named execution plan. In contrast to stored functions, stored procedures are compiled once and used endlessly without having to be compiled again.
Advantages of stored programs [2]:

- **Enhancing Database Security**

  In the event that an application account is compromised (for instance, the password is “cracked”), the attacker will still only be able to execute the stored programs, as opposed to being able to run any ad hoc SQL. A programmer can ensure that these SQL statements are surrounded by whatever business rule validation or logging is required. While such a situation constitutes a severe security breach, at least the programmer is assured that the hacker will be subject to the same checks and logging as a normal application user.

- **Improving portability of the application**

  Using stored programs can help avoid RDBMS-dependent code in application Layer. In theory, stored programs called by different databases can be made to look and behave identically from the application’s perspective. Of course, the underlying stored program code will need to be rewritten for each RDBMS, but at least the application code will be relatively portable.

- **Improving maintainability and efficiency of the application development**

  Changing application code several times is not needed. Also, application development efficiency may be enhanced if the database developers are able to implement the data access routines directly in DBMS using the stored program language.

- **Decreasing the traffic of network**

  If the application code is written entirely outside the database, accepting input from the end user, reading some data in the database, deciding what statement to execute next, retrieving a result and the like would require a network round trip between the database and the application. The time taken to perform these network trips can easily dominate overall user response time and performing the calculations in a stored program will reduce network overhead, which might reduce overall response time.

- **Common Routines Across Multiple Application Types**

  It is not at all uncommon for multiple applications to share a single database. Implementing common code in stored programs may allow these applications to share critical common routines. For instance some versions of the stored procedure contain code for handling deadlocks and an optimistic locking strategy that perform a transactional-safe and, logged transfer of funds between two accounts.
Mechanism for Data Abstraction

The use of stored programs in development provides a convenient way of implementing data access layer. It is a good technique to separate data access code from business logic and presentation logic. With data abstraction, the impact on higher-level logic will be minimized when there are changes to the underlying data structures.

Disadvantages of stored programs [2]:

- **Slower for computationally expensive operations**
  
  In general, DBMS stored programs, are slower than languages such as PHP, Java, and Perl when executing “number crunching” algorithms, complex string manipulation, and the like.

- **Not supporting Mapping Systems**
  
  Most object relational mapping systems such as Hibernate framework in java does not support using stored programs.

- **Sometimes more portable than native SQL**
  
  It was said earlier that stored programs could be used to build RDBMS-independent applications by encapsulating RDBMS-dependent SQL in stored program calls. Unfortunately, this is only possible for RDBMS types that support similar semantics for processing parameters and returning result sets. Oracle is an exception in this regard; So applications which can use only stored programs are reasonably portable between MySQL and either DB2 or SQL Server.

- **Difficult to find bugs**
  
  They can lead to atomization of the application logic and make it difficult to find implementation bugs and design mistakes.

- **More Skills Needed**
  
  It is needed to have more skills in development team to write program in this language.

- **Difficult to debug**
  
  Debugging application errors that involve interactions between stored program code and other application code may be many times more difficult than debugging code that is completely encapsulated in the application layer. For instance, there is currently no debugger that can trace program flow from the application code into the MySQL stored program code.
Stored procedures features

A store procedure allows modular programming because it can be declared once, and can be called any number of times in a program. It balances processing load carefully between a client and a more powerful server machine and using it is one method to reduce the load on the client, which might otherwise be overloaded. Also, Execution of multiple server-side operations in a single place such as stored program is faster and can reduce network traffic which is often a serious constraint on client-server applications. One of the specifications of stored procedure is tight integration to SQL and it does not need to use ODBC or JDBC for executing SQL statements [2]. The other one is centralizing some of the processing on the server which allows a greater measure of control over core logic because maintaining correct versions of client software is often problematic. In addition, these procedures can have security attributes (such as permissions) and ownership chaining, and certificates can be attached to them. They apply restriction on the database account only to well-defined procedure calls, rather than allowing the account to execute all SQL statements.

MySQL is a platform of database systems that is originally from Unireg which is an in-house database tool. Unireg is a tty interface builder that uses a low-level connection to an ISAM storage with indexing and was used by the Swedish company TcX. Some interfaces were added to Unireg and the first version of MySQL was created [2]. Then, the other versions were produced one by one with some improvements. At the end of 2005, MySQL distributed a function based version that works in commercial RDBMS systems. In this version, functions, stored procedures and triggers are introduced.

Concept of Optimization by Stored procedures

In addition to adding application functionality to programs and efficiency in developing by stored procedures, it can do other things that non-procedural language cannot do. For example, how to retrieve data could not be specified in nonprocedural queries, because it just depends on MySQL itself to determine how to go about identifying the result set. From time to time, a very good idea might be found about the most efficient way to retrieve the data, but the MySQL optimizer might choose another-less efficient-path. When a DBA thinks he/she knows how the data should be retrieved but can’t get the optimizer to play ball, using MySQL stored programs can sometimes force the desired approach [2].

The following questions are addressed in system performance:

- Do all stored procedures make performance better or not?
- Which kinds of stored procedures are able to optimize the queries and which of them are not?

These questions lead the programmer to check some functionalities of the system such as time measuring and internal optimization of stored procedures.
Time measuring

Comparing the execution time of stored procedures to other SQL queries is a method that can be tried. For example, the performance was compared in two sample programs that do equivalent complex mathematic operations and the time specific is measured. One of them calls stored procedure and the other one uses query statements while the same SQL operations exist in both of them. The results are shown below:

**Figure 8: Query statements vs. stored procedure for doing complex operations.**

In some cases such as computation of complex arithmetic and mathematically large operations, calling stored procedure is slower than in an equivalent SQL statement because mathematically intensive operations is performed poorly and should be avoided. Apart from these, splitting complex SQL statements into independent sequence of small queries is a method that can be tried to optimize running time of these statements.

There is another diagram that shows the results of two sample programs in remote machine vs. local machine. According to the diagram, if these programs are in a local machine, the data are not needed to be transferred across the network and the time consumed in this state is even less than that for calling stored procedures. On the other hand, there are special differences between the time of calling a stored procedure and sending query statements in a remote machine caused by network traffic. In the latter, the program has to send some SQL statements and fetch result rows across the network which performing these operations takes much longer to execute [6].

**Figure 9: Diagram of execution time in a stored procedure and equivalent query statements in remote vs. local machine**
Internal optimization

Sometimes there are some SQL statements that become complex because of limitations in SQL syntax. MySQL optimizer cannot make a good plan for these limitations [2]. Some of them are described below:

- **Expensive self-join**

  This happens when a query is constructed to join a table to itself that is used in order to filter the specific rows. MySQL does not provide the SQL syntax that allows us to return this data without an expensive self-join. A stored procedure retrieves data in a single pass through a temporary table which can later show the results.

- **Correlated updates**

  This is an update statement that consists of a correlated sub query in the set clause or where clause. For example, correlate update happens on a table which must be used twice in a query, because a stored procedure identifies the rows of a table by using a cursor and an update is issued for each retrieved row by the cursor.

There are also some features in the syntax of stored procedures that should be used in a right way. Some of them are shown below:

- **Cursor**

  For the CURSOR with returning one row in result set, it is better to use INTO instead of fetching the cursor [2].

- **Loop**

  To use LOOP, it is better to make ‘leave’ or ‘continue’ to avoid needless processing and unnecessary statements out of the loop.

- **Case versus If**

  CASE is much better than ‘IF’ for the condition that one variable is against a range of set values.
Structure of stored procedures

A stored procedure is a standard database object that can be manipulated by using standard DDL statements such as CREATE and DROP. Some parts of a stored procedure default structure are explained as follows.

First of all, a blank script file is created in MySQL query browser. Since default delimiter in SQL is ";" (Semicolon) and it is used at the end of query statements, this delimiter should be changed into another delimiters except semicolon such as "$$". Also any procedures with the selected name must be dropped [2].

```
Delimiter $$
Drop procedure if exists (name of procedure)
$$
```

Now, a stored procedure is created with an arbitrary name and it can accept parameters. These parameters have three types that consist of input, output and INOUT attribute. INOUT attribute can be used as an input and as a return value by the procedure.

```
CREATE PROCEDURE ` (name of procedure) ` (IN game_id INT(10), OUT user_name varchar(11),INOUT date DateTime)
```

Then, a BEGIN END block is used for the procedure and all procedure statements are written into this block. After finishing stored procedure, "$$" delimiter is used to show the end of procedure and the delimiter is returned to ";". Next, the variables used in the procedure should be declared. These variables have different types similar to attributes in tables (int, varchar, date, double, etc). The query should be created by using reasonable syntax. To do this, if the query is an insert, update or delete, it works the same as a SQL query. If it is a select operation, it depends on the number of rows returned by the result set. If they are not more than one, it should be set into user-defined variables in procedure.

```
Select name into username form users where id = 1;
```

For more than one result, a cursor should be declared to trace a loop defined in procedure. All variables needed are defined in procedure before creating statements.

```
Declare cur1 cursor for select name form users where id <10;
```

After declaring the cursor, a handler can be implemented for checking whether the rows in result set are finished or not.

```
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done=1;
```

Then, the cursor is opened in a loop with an arbitrary name and the corresponding codes are inserted into the loop block. Now, the rows should be fetched into the user-
defined variables. Then, a condition should be declared to make sure when the number of rows finishes, the loop will be left and the cursor will be closed.

```sql
OPEN cur1;
cur_user: LOOP
    FETCH cur1 INTO username;
    Call 'another procedure' (username); (e.g.)
    IF done=1 THEN
        LEAVE cur_user;
    END IF;
END LOOP cur_user;
CLOSE cur1;
```

There are also some kinds of loop in stored procedure programming:
- Simple loop statement which has leave statement and iterate statement (used to restart execution at the beginning of a loop)
- While loop
- Repeat until loop

Apart from the previous syntax, most of stored procedures use IF-ELSE or CASE condition to handle queries. In addition, every stored procedure can be called by other procedures.

**Showing result set in a stored procedure**

To show the result set from a stored procedure, it is a good idea to use temporary table for storing data. To insert data in a temporary table, each row of select operation is assigned to a variable and this variable is inserted into this table.

```sql
CREATE TEMPORARY TABLE IF NOT EXISTS 'temp_table_name'(parameter1 INT(11),parameter2 varchar(11));
insert into temp_table_name (parameter1,parameter2) values (p1 ,p2 );
select * from temp_table_name;
```

**Stored procedure in different vendors**

Stored procedure languages are quite vendor specific which is one of the main disadvantages of this language. To switch another vendor, overwrite the stored procedure must be performed. Oracle, Microsoft SQL server, POSTGRESQL and MySQL are different vendors that support stored procedures and the syntax of writing these procedures are different by vendor. Some of them have more features and provide a better environment than others.

Supported tools for writing or debugging stored procedures in different vendors are not as much as they are in different programming languages. A database admin (DBA) normally writes a stored procedure and every vendor improves specific kinds of performance. So, the performance difference between stored procedures may differ by vendor because compilation speed and the skills of DBA in each vendor have effects on it.
Chapter 4

New version of database in Planet Mule-2

The new version of database for Planet Mule-2 has some added and changed features according to the information achieved in the previous sections. These features are shown below:

Adding / deleting some tables

- Some tables like colony_event or personal_event are new in the new version. These tables are for the events which happen respectively for a game and for a player in a game. Both of them might happen on a round of the game and affect round_results table. The personal_event table includes the id of game and user number in the game, while the colony_events table just includes the id of game.

- Also, some tables from the game should be deleted, because they are not needed in the new version of the game. These tables are planet_maps, event_times, planet_map_coords, lobby_chats, and badges.

Modifying attributes

Modifying attributes means altering some columns in some tables. For example:

- There are some columns in games table that are not used in the new version such as water and development and host_user_id. The main reason for deleting host_user_id is setting up a new host computer for new version of Mule game and there are not any users to be hosted.

- There are also some attributes added to this version such as chats_path_name to games table and logs_path_name to users_games table caused by merging chats table to games and logs table to users_games. The top_users table is extended to collect the information of players in every week instead of last week. So, two properties of week_number and id are added to this table.

- There are three columns with a varchar data type that repeat many times in six tables. It is better to reference to them with a number. Making new table is not efficient because the main table has to join with this table for getting the column value. So, the column is modified to be a data type of Enum as a map of id and string value in the database. These columns are as follows:

  1. race in user and round_results tables
  2. mode in ranks and user_statistics tables
  3. event_type in colony_event and personal_event tables
Creating Relations

The relations of information contained within the tables are created for the new version of the game. These relations cause some constraints on the tables that can prevent a potential wrong insert or update. These constraints include domain, entity, referential and foreign key integrity constraints which limits the application in doing CRUD operations in a specified range. For example, to choose a primary key, the uniqueness should be checked by domain integrity. In addition, having proper value for the primary keys is checked by entity integrity constraints. Deleting/changing a record/primary key from a primary table cannot be done if matching records exist in a related table. Also, entering a value except null is not allowed in the foreign key field of the related table that does not exist in the primary key of the primary table by referential integrity. Finally, if a referenced row in a table is updated/deleted, its children should be updated/deleted as well by foreign key integrity constraints.

Merging some tables with one-to-one relationships

There are some tables that have one-to-one relationships with other tables for the most of rows. Merging these tables is preferred because one-to-one relationship is used when there are many null values in a table. For instance personal_events table has one-to-one relationship with round_results table, but they cannot be merged with each other because personal events are just filled for the affected players and for the other players remain null and too many null values are stored. Nowadays, according to the large amount of memory used for storing data, the cost of memory is not that much. The merged tables of the new version are described as follows:

1) **USERS vs. SKILLS**

![Diagram of USERS vs. SKILLS](image)

**Figure 10: Relation between Users and Skills tables**

Most users who play the game have skills. So, it is a good idea to merge them to each other to avoid storing many null values. [21].
2) **USERS_GAMES vs. LOGS**

![Diagram showing the relationship between Logs and Users_games tables]

*Figure 11: Relation between Logs and Users_games tables*

`users_games` and `logs` tables can be merged because there have to be a log for each player in a game. It might store many null values, but with high speed databases, this approach will not have a significant impact on performance and it is recommended [21].

**Applying Normalization Form**

**Update Anomalies**

An Update Anomaly exists when one or more instances of duplicated data are updated, but not all. For example, consider Ali changes his name. It is necessary to update all instances of Ali’s name.

<table>
<thead>
<tr>
<th>Game_id</th>
<th>User_id</th>
<th>User_Number</th>
<th>Player_index</th>
<th>Name</th>
<th>Disconnected</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Ali</td>
<td>false</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>Ali</td>
<td>false</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>Frank</td>
<td>false</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>Richards</td>
<td>false</td>
</tr>
</tbody>
</table>

In the old version of *Planet Mule* there are two redundant columns in `users_games` table that by solving these redundancies, too many redundant rows in this table are deleted (1NF). These are mentioned below:

One of them is `name` column which shows the name of user and is not located in a right place. The `name` column which includes the name of all players except AI players can be deleted from `users_games` table and put in `users` table. There is an AI player in `users_games` table that can be used as substitute player to supplement the number of players. To do this normalization, five constant names can be set as AI players in the application code according to their ids in the database (id:-1,-2,-3,-4,-5)
The other one is player_index that is used for showing spectator role in Users_Games table. Deleting player_index attribute affects user_number column in this table and can be done by adding some possible extra id number (such as -1, -2, -3 …) as a substitute for player_index in user_number column for spectators.

Delete Anomalies

A Delete Anomaly happens when certain attributes are lost because of the deletion of other attributes. In games table, the mode column is string and if one row with a specific mode type is deleted and the other rows do not have that value, this mode type will be lost, such as mode_2 that is just for Frank in below table.

<table>
<thead>
<tr>
<th>Game_id</th>
<th>Host_user_id</th>
<th>Name</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Ali</td>
<td>Mode_1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Ali</td>
<td>Mode_1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Frank</td>
<td>Mode_2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Richards</td>
<td>Mode_3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Richards</td>
<td>Mode_3</td>
</tr>
</tbody>
</table>

Insert Anomalies

An Insert Anomaly is the opposite of delete anomaly and occurs when certain attributes cannot be inserted into the database without the presence of other attributes. For example a game with a new mode cannot be added in the table below, unless at least another game has that mode. Since there is not any row with Mode_3 value, inserting a game with that mode is not possible. To solve this anomaly, the mode column values should be inserted in another table with new ids and their ids are referenced by the games table. If the number of values for the mode column is not so much, using Enum data type is preferred.

<table>
<thead>
<tr>
<th>Game_id</th>
<th>Host_user_id</th>
<th>Name</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Ali</td>
<td>Mode_1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Ali</td>
<td>Mode_1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Frank</td>
<td>Mode_2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>Richards</td>
<td>Mode_4</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Richards</td>
<td>Mode_4</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>Michael</td>
<td>Mode_5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Richards</td>
<td>Mode_6</td>
</tr>
</tbody>
</table>

Apart from these anomalies, in Planet Mule, every round has a start time for each user in a specific game. So, in the Mule Database the round_results table has a time column for each record. Since the value of this column should be the same for all users in every game, it is better to use this column in another table with a composite
key (2NF). The *time* column depends on part of *round_results* primary key *(game_id, round)* and can use this key as a primary key. Apart from this, all users do not require the start time of every round and the *time* column can be deleted from the table.

**Adding indexes**

Indexes for the keys of tables automatically are added, whether they are foreign or primary. It is a good idea to add index for every attribute that is searched many times in a large table. For example, it is recommended to insert an index for *disconnected* attribute which indicates the player’s status in *users_games* table. Since, many users normally leave during the game due to lack of time or disconnection, *disconnected* attribute is changed very much.

**Designing an ER-Diagram**

ER Models are represented by ER Diagrams. An entity and its attributes are bound to a table and its columns in a database. Similarly, a relation of two entities is bound to a relation of the corresponding tables. A sample is described below:

![ER-Diagram](image)

*Figure 12: A sample of ER-Diagram*

**Enhanced Entity Relationship Diagram for Planet Mule-2**

The tables used in the new version of database are mostly optimized and free of redundant data. These are nine tables whose relations are optimized as well and are explained below:

- **USERS**
  This table has a relationship with *members* table in *muleforum* database and consists of information of users such as *forum id*, *last alive*, *last login*, and *race*. 

37
Figure 13: ER- Diagram of the new database designed

- **GAMES**
  This table consists of the past games, whether those games are finished or not. Attributes of this table are name of the game, number of spectators, number of rounds played, and the path name of chat file related to the game.
• USERS_GAMES
   This table consists of the information of every user in every game and contains some attributes such as score and colony.

• ROUND_RESULTS
   This table includes the data of every round for every user such as round number, money, goods, plots, total score, personal event, colony event, race name, etc.

• RANKS
   This table keeps the ranks of every user in all games against other users according to number of winning games. It is covered by user id, mode, rank, etc.

• TOP_USERS
   This table shows the top users every week for uploading on the website. It has user id, game won, week number columns.

• USERS_STATISTICS
   This table keeps the statistics for every user. It includes personal score, colony score, games finished, games legally finished, games abandoned, etc.

• COLONY_EVENTS
   This table shows the colony events happening in every game. It can reduce or add score to one or several players according to the type of event and is covered by game id, event type, money, etc.

• PERSONAL_EVENTS
   This table mentions the personal events which happened for every user in a game which affect one player in the game with reduced or added score. It consists of user number in the game, game id, event type, money, etc.

Choosing optimized method for using queries

Some solutions were proposed to choose the efficient way for using queries. Firstly, the time was measured for stored procedures and equivalent query statements in different states such as local host machine, remote host machine and complex arithmetic queries. Secondly, some important features for stored procedures such as security, modular programming, vendor specific, and difficulty to debug were considered. Selecting an efficient query should be done in accordance with the programs’ state[23] [6].

In Planet Mule, there is a central server where the application runs on. Another server exists where the database resides. The connectivity between these two servers is like a remote host machine. So there is much time difference between calling stored procedure and equivalent query statements by the application. Since there are two servers for the Planet Mule-2, stored procedures can decrease network traffic at the minimal level and optimize performance.
In addition, different operations may be used in queries and the security can be implemented in different places which can affect the response time. So, calling stored procedures for a large number of queries and using query statements fewer times is preferred. Apart from this, using stored procedures adds some functionality such as security, faster execution and being modular to the program which affects execution time.

**Stored Procedure Code Sample**

There was just an application code written by PHP Programming language in the old version of *Planet Mule* that is necessary to convert to stored procedure. The stored procedure which is called after finishing every game is described below:

```sql
CREATE DEFINER=`root`@`localhost` PROCEDURE `finish_game`(IN gid INT(10), IN uid INT(11), IN fid boolean)
```

These are the Properties of the procedure that can be IN, OUT, INOUT

This is the Name of stored procedure
BEGIN

declare l_userid,l_playerIndex,l_legal,numHumans,done INT(11) default 0;
declare l_disconnected,l_mode VARCHAR(20) default 'false';

DECLARE CURSOR FOR SELECT user_id, disconnected, player_index FROM users_games
WHERE game_id=gid;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done=1;
SELECT mode into l_mode FROM games WHERE id=gid AND host_user_id=uid ;

OPEN cur1;
cur_user: LOOP
    FETCH cur1 INTO l_userid,l_disconnected,l_playerIndex;
    IF (l_userid!=0 and l_disconnected !='1' and l_playerIndex >= 0) then
        set numHumans = numHumans + 1;
    end if;
    IF done=1 THEN /* No more rows*/
        LEAVE cur_user;
    END IF;
END LOOP cur_user;
CLOSE cur1;
if (numHumans > 1 ) then
    set l_legal = 1;
end if;

UPDATE games SET status='finished', legal=l_legal, failed=fld WHERE id=gid AND
host_user_id=uid;
call updateTopUsers();
call updateRanks(false); 
call updateHostScore(uid); (Omitted)
call mulevent('games_changed'); (Omitted)
call updateSkills (gid); (Omitted)
call updateSkillRanks();(Omitted)
END
A: Application send request to finish_game.php

send request to finish_game.php

PHP files

Database

The first query (from games)

The second query (from users_games)

The third query (update from games)

updateUserStatistics()

updateTopUsers()

muleEvent()

updateRanks()

updateHostScore()

updateSkills()

updateRankingBasedOnSkills()

commit()

return status

write audit log()

return status

B: Application

Database

After finishing the game

Call procedure finish_game()

Return status and response to application

Figure 15: The time spent in a part of project to use queries in comparison to stored procedures (A: Query statements, B: Stored procedures)
Java interface for calling stored procedures

Stored procedures must be called by the java application. First, the application uses a connector named JDBC to create a connection to the database. Then, it uses the `preparestatement` interface for calling procedures and executing query statements. `Preparedstatement` is an object that represents a precompiled query statement. This interface with their parameters is executed and makes a result set which is returned to the program. Some parts of the application code which call stored procedures in the program are shown as follows:

```java
private Connection conn = DbManager.getUniqConnection();
public void sendFinishGame(int gameId, int userId, boolean failed) {
    try {
        String sql = "call finish_game(" + gameId + "," +
                     userId + "," + failed + ")";
        PreparedStatement pstm =
                        conn.prepareStatement(sql);
        ResultSet rs = pstm.executeQuery();
    } catch (SQLException ex) {
        Logger.getLogger(Database.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Conclusion and future work

A well-designed and responsive database was created to take efficiency into account and keep redundancy at minimum level. In addition, the existing queries were needed to be efficient.

To improve the database, some tables which had redundant data, were fixed by applying database normalization techniques. Also, some tables which had one-to-one relationships to other tables were merged together in order to minimize the number of relations between tables. To decrease the number of joins created between tables, an Enum data type was used instead of a small table with single id and value. Next, the columns of some tables were deleted and their values were inserted into other columns in order to include the required values. Moreover, an index was added for an attribute which was searched many times in a large table. Finally, to visualize the implementation, an ER-Diagram was designed.

The logic of grouping queries reduces the network round trip between the database and the application. On the other hand, in a remote host machine, the response time between a sample stored procedure and equivalent queries was measured. As a result, stored procedures were quicker than query statements. Stored procedures also were optimized internally to achieve a lower response time.

For this game, the company plans to create a website by PHP to show the latest scores. Implementation of this database using NoSQL databases could improve performance further. A new design addressing this database is suggested for future work.
References

[2] MySQL Stored Procedure Programming Building High-Performance Web Applications in MySQL by Guy Harrison, Steven Feuerstein