Final Report

Rate my world leader

Adam Finer, Thomas Bonatti, John Hamilton

2014-2015

**Table of Contents**

**Table of Contents1**

**Executive Summary2**

**Introduction2**

**Problem Description3**

**Solution Description4**

Front End4

Back end4

**Key Challenges5**

**Database Design6**

Integrity Constraints6

Stored Procedures6

Views7

Indexes7

Triggers7

**Design Analysis8**

Strengths8

Weaknesses8

**Appendix9**

**Glossary11**

**References12**

**Index13**

**EXECUTIVE SUMMARY**

This final report document begins with a description of the initial problem that this project was intended to solve. The solution to that problem, a discussion of the key challenges that were faced over the course of the project, and a description of the database design will follow. The description of the database design will include the integrity constraints that were applied, the indexing scheme, views and stored procedures that were used, and a discussion of the pros and cons of the design.

**INTRODUCTION**

This document is the final report on the Rate My World Leader project developed by a team consisting of Adam Finer, Thomas Bonatti, and John Hamilton. The purpose of this document is to properly and effectively describe the problem described in previous documents, the Final Problem Statement, and to additionally describe the solution implemented by the team.

**Problem Description**

In the modern world, there are numerous people from hundreds of different areas who are considered to be world leaders. There are presidents, prime ministers, kings, queens, senators, governors, and numerous other positions that those leaders fill. Taking the concept of the Rate My Professor website of rating professors on a scale and commenting on them, the team envisioned an idea to create an application that would allow users from around the world to rate the leaders of the world. The end result of this program was intended to be a user interface called Rate my World Leader on which a user could login and use the implemented features of the application.

Below is the list of features for the final version of the database, there are some additional features that have been added on since the final problem statement was turned in.

|  |  |
| --- | --- |
| Feature Name | Feature Explanation |
| User accessible application | The way to access the application is through a downloadable application |
| Administrative privileges | If a user is of administrative status, he or she can delete non admin users, add and update the status of world leaders, upgrade non-admins to admin status, and many other actions that normal users can not perform. |
| Users | Any person can access the application and create an account to access the application interface |
| Rate leaders | Any user, once they are logged into the database, can add a single rating for any leader. The rating consists of a scalar from 0-10 where 0 means they hate the leader and 10 means they believe that leader is the best leader on the planet and a description of the leader. |
| *Game of Thrones* References | We reference characters and places from the *Game of Thrones* universe |
| Leaders | A list of some Major World leaders is include so that the current user database (just the three developers and some dummy users) can rate leaders |
| List of Locations | To allow for possible future development, a list of locations including all of the Continents, Countries, all of the US states, and a few cities were created. As of now, not all of those places have leaders (only countries do due to the scope the team decided to go with). |
| Political Parties | A list of political parties for leaders is included for additional information about world leaders |

**Solution Description**

The solution that the team decided upon was to develop an application using java graphics and Microsoft SQL server Management Studio 2012 to store the information related to the database.

## Front End

Using Java Graphics, the front end was created which represents the Graphical User Interface (GUI). Each window was designed to only contain the bare minimum of components, consisting of only a label to say what page the user is currently on, the buttons that page needs, and any text fields or tables needed to allow the GUI to function. Upon opening up the application, the user has the option to create a new user, log in, or close the application. Upon logging in, the user is taken to the User page, which gives the user the choice of logging out, going to options, or searching for a user/leader. In the options menu, the user can delete their account, or change their password. The search menu brings up a window that has three text fields and the option to search via leader, or user. If leader is selected the three fields stay, counting for first name, middle initial, and last name, respectively. If User is selected, the bottom two text fields disappear and the topmost is for username. Upon typing in the respective search, the application checks the database to find said leader or user, and then takes the current user to the page of the searched leader or user. This page contains the Leader’s ratings, or the ratings the user has made, respectively. If a user has not rated a search leader, the buttons and text field to rate that leader are shown as well as the existing ratings for that leader. The GUI and application also sanitise inputs to prevent SQL injection, making the database much more secure.

## back End

The back end of the solution, which was created using the SQL server software, contains the data the front end GUI uses. This data base contains the seven tables: Leader, Leads, Location, PartyMember, PoliticalParty, Rating, and User\_Account which interact with the user interface to enhance the user experience of creating and browsing world leaders. This database currently exists on the server titan.csse.rose-hulman.edu provided by Rose Hulman. Once this server is shut down though, it is still unknown where the database will be stored.

**Key Challenges**

Challenge: Little experience using SQL Server

Solution: For most problems, syntax could be looked up online, because of the extensive website provided by Microsoft, or in the lecture notes from class.

Analysis: The internet was useful for answering all of the questions. But some issues took a while to resolve because some of the websites were difficult to read and understand based off of the knowledge of databases the team had.

Challenge: Team Coordination and Data Sharing.

Solution: A git repository was created for the team to coordinate and share work.

Analysis: The Git repository turned out to not be very useful. SQL server stored queries written, and documentation was mostly needed at the beginning and end of the project. During the project, a dozen ASCII file SQL queries were committed to the repository, which only cluttered it because SQL server stores the queries anyway. Some of these queries are out of date because functions defined on the database were altered at a later date and the queries on Git were never updated.

Challenge: Balancing Team Work Load

Solution: The team was unable to resolve this challenge in the time frame of the project. But for future reference, they will have learned from this experience with others and work as better teammates.

Analysis: Overall this challenge was handled poorly. Members of the team had other time commitments related to other classes and did not balance their time well. Team meetings were often last minute, and poorly communicated by all three team members.

Challenge: Making a GUI takes a lot of time and focus. And tends to take a long time.

Solution: Work on the whole project was mostly split into the Database side, and the Application side. So that those working on the Application side would not need to worry about the database and vice versa. John Hamilton was largely responsible for the application, while Thomas Bonatti and Adam Finer focused moreso on the database side.

Analysis: This approach worked mostly well, the GUI ended up taking a longer time than expected, making the amount of work done a little unbalanced, however, because of the freedom that came with not having to worry about the other end of the Application as a whole allowed the work to get done efficiently and effectively.

**Database Design**

## Integrity Constraints

If you look at the Entity Relationship Diagram in the Appendix section of the report, you will be able to find the different foreign key and primary key constraints of the database. You can find a detailed description of those constraints in the Security Analysis.

Listed below are the additional constraints on the database:

* Emailcheck on the USER\_ACCOUNT table checks the email attribute to make sure the emails are considered to be valid emails (definition can be found in the glossary)
* In the procedure, before a password is hashed with its username using SHA1, it is checked to make sure it is greater than 7 characters
* The username is required to be at most 10 characters long and to not be NULL
* Many various attributes such as username are not allowed to be null
* The rating on the Rating table is specified in the addRating sproc to not be allowed to outside the range from 0-10
* Only administrators can perform such actions as deleting an account that is not associated with their username
* As specified later on in the trigger section, there is a trigger on the party member and leads table which forces there not to be any overlapping of dates for different situations
* On the leader table, gender can be one of two characters, M or F, representing the genders of Male and Female
* On the location table, only certain types are allowed in the table (i.e. Continent, Country, State, City) and any others will be rejected

## Stored Procedures

|  |  |
| --- | --- |
| Stored Procedure name | Procedure Purpose |
| Add Rating | Allows the user to properly add a rating on any leader of their choosing |
| adminaccount | Allows an admin to make any normal user into an administrator |
| changepassword | Allows an user to change their password as long as they have their old password |
| CreateLogin | Allows someone to create a non-admin user account with a valid email, username, and password. Through this procedure, the password given is Hashed and saved in the database. |
| Delete\_account | Allows a normal user to delete their account which also deletes any ratings they may have created. |
| loginsp | Allows a registered user to log into the application. |
| RemoveAdminPrivlage | Allows an administrator to remove admin privileges from another administrator. |
| RemoveRating | Removes the specified rating, given by the primary key of Rating. |
| usersratings | Creates a table of all the ratings for a given user. |

## Views

The Rate Your World Leader database contains the following six views:

* dbo.AllLeaderRatings
* dbo.AverageLeaderRating
* dbo.CountryView
* dbo.PartyMembers
* dbo.PublicUserView
* dbo.UserRatings

These views were used to monitor certain features of the database during its creation. For example, the CountryView was used to see what countries had already been added to the database so that we did not add any repeats. Another use for the views was to create viewing pages for the application. For example, the UserRatings view can be seen from a window in the application. Generally, views were not necessarily needed for this particular project, because most of tables seen in the application are created by select statements and stored procedures.

## Indexes

There was only one developer created index. A nonclustered index was created for the Ratings table called useraccount which was created because there are searches done on the database by user for searching for ratings by username. Clustered indexes were pre-created by Microsoft SQL Server Management Studio on all tables based on their primary keys. If this were not the case, clustered indexes would have been created on the USER\_ACCOUNT table based on username, the LEADER table based on leader\_id, and on the LOCATION table based on the ppID for each location in the table.

## Triggers

The following triggers are listed below with explanations for why they are in the database.

1. insertLeadsTrigger- the intended purpose of this trigger was to prevent any leader from leading the same location at the same time as another leader and to prevent a leader from leading two places at once at the same time.
2. insertPartyMemberTrigger- Similar to the previous trigger, this trigger was intended to prevent, on insertion, for a leader to belong to a single party at the same time that he or she belongs to another given party.
3. CreateRating- This trigger was intended to properly insert a rating in such a fashion that the time stamp is correct for the rating. Also, the trigger increments whatever user created the rating’s number\_rating attribute in the USER\_ACCOUNT tables by 1.
4. DeleteRating- this trigger does the opposite of the previously defined trigger, accept on delete instead of insert
5. DeleteAccount- this trigger deletes the tuple specified in the deletion statement as well as deletes that user’s ratings from the Ratings table to keep integrity constraints sound.

# **Design Analysis**

## Strengths

* User passwords in the database are hashed with the username of whatever user that password belongs to allowing for enhanced security of user passwords
* Many integrity constraints are in place to allow for foreign keys, likeness for valid emails, and minimum lengths for usernames and passwords
* Table, trigger, sproc, attribute, and view names are appropriate and properly describe their functionality
* There are constraints in place to allow administrators special privileges
* The location table contains many places around the world and from the *Game of Thrones* universe
* All the developers of the project can access the sources behind the user interface and the database at any time. They can also view all of the documents associated with the project through git hub
* SQL injection is prohibited through the GUI code

## Weaknesses

* Users are unable to view their passwords. With the current state of the database, a user who forgets their password will not be able to reset or change their password. Resulting would be additional data in the database which consists of inactive users. What could be done is a policy which states that a single email can only have one username associated with it or deleting an account which has not created a new rating in a certain amount of time.
* The team did not import or create mass amounts of data, so there are very few tuples in the database. This results in the inability to demonstrate some of the implemented features of the database, like showing the view titled UserRatings will not show as much data as it would if the database was full of ratings.
* The team could not figure out how to delete unneeded indices that were on the tables. So when they created the nonclustered index on the Ratings table, they were unable to delete the unnecessary clustered index that was created by the database itself.
* Too many attributes allow for NULL values because too much data was already inputted into the database before this fact was noted.
* Not many update triggers are not included in the database because data updates should not mess with the integrity of the database

**Appendix**

## Relational Schema

**Leader**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Leader\_id | gender | bday | fname | lname | midint |

**Leads**

|  |  |  |  |
| --- | --- | --- | --- |
| leader\_id | startdate | location\_id | enddate |

**Location**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ppID | superppID | type | name | population |

**PartyMember**

|  |  |  |  |
| --- | --- | --- | --- |
| leader\_id | startdate | party\_id | enddate |

**PoliticalParty**

|  |  |  |
| --- | --- | --- |
| ID | placeOfOrigin | party\_name |

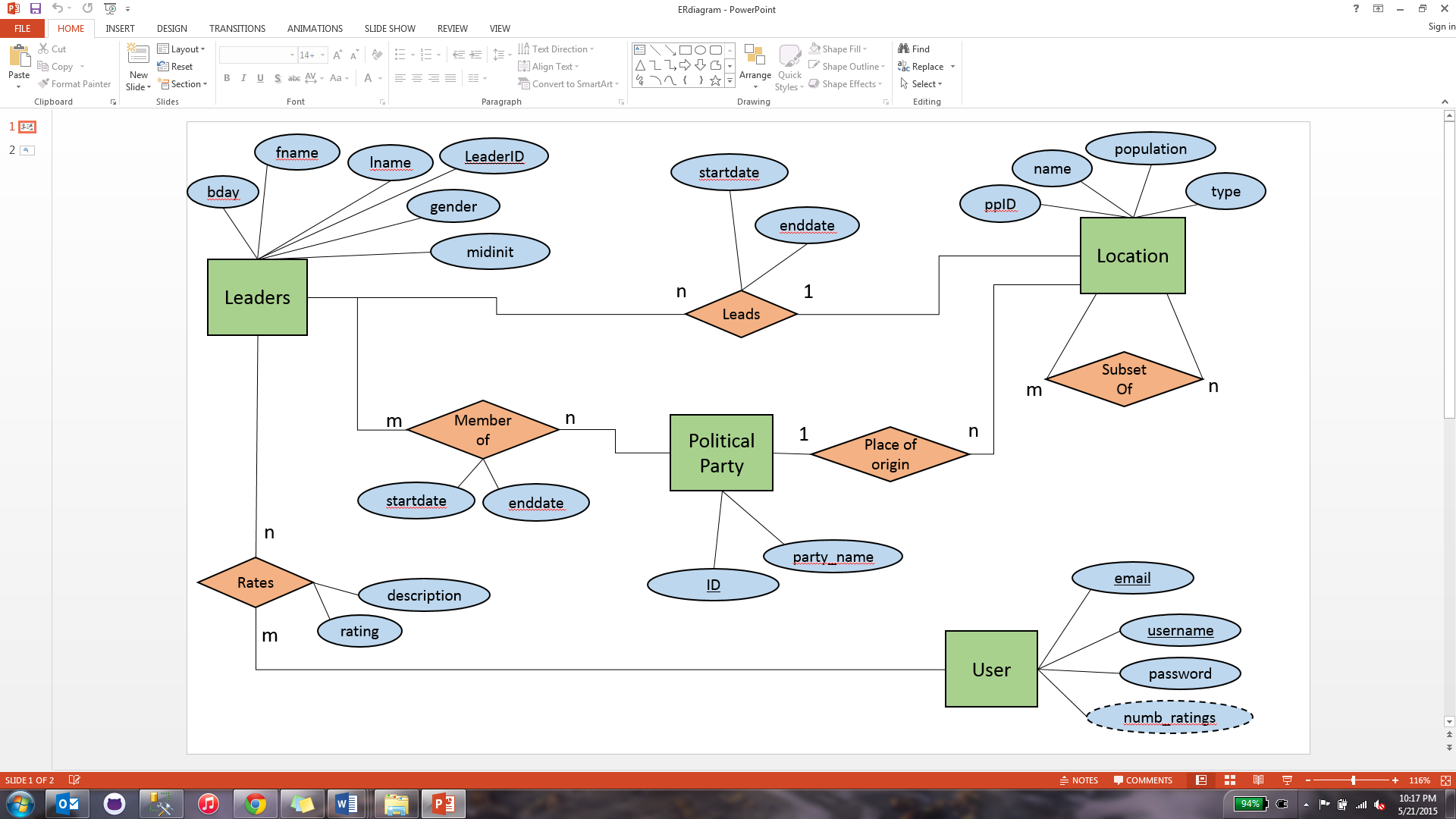
**Rating**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| leader | username | rating | text | date |

**UserAccount**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| username | email | number\_ratings | passwordHash | isAdmin |

## Entity RelationShip Diagram



## Description of Entity relationship Diagram

The Key points of the ER diagram are the user account and Leader Entity sets. Leaders, found in the upper left corner are main focus of the project, and they are what is rated by users in the database. The leader ID is a numerical identifier unique to each leader, and the leaders name, gender, and birthday are also stored. Equally important is the User Account entity set in the lower right corner. The User Accounts are important as a means for users to rate leaders, and to avoid having anyone place more than one rating on a leader, since they need an account to make ratings. User Account stores unique usernames and emails for each user, as well as the SHA1 hash of the user’s username and password concatenated together, to ensure the security of their account. The location and Political Party entity sets are provided to give users extra information by which they can filter the leaders they see.

**Glossary**

Game of Thrones: a television show/ book where some of the locations, leaders, and users are from

Git-hub: a publically available application that stores the documents and application for the project

GUI: (graphical user interface) this refers to the display on the screen that users interact with to manipulate and search the database

Index: For a database, this is a data structure which can be navigated through to access and search data in a quick fashion than just searching through a table tuple by tuple. These are especially useful when using a large amount of data or when accessing certain through one or more attributes of a table.

Integrity constraints: a constraint imposed on a database to maintain certain relationships in the database such as a certain likeness for data, foreign key relationships among tables, or uniqueness constraints

Query: sql code which describes a certain task to be done by the database such as creating a sproc, creating a trigger, or manipulating the database in some other way

Rate My Professor: a website where students can rate their college or university’s professors and search for other ratings

SHA1- hashing algorithm used on the database

SQL: a language used by the Microsoft SQL server database management software program to write queries to manipulate the database

Stored procedures: (sprocs) procedures in the database which can be executed to perform certain tasks like adding data in a certain way or accessing a certain set of data.

Trigger: similar to a sproc, this is applied to a table in the event of an insert, update, or delete on a that table. It can occur before, during, or after those previously defined actions.

Valid email: and email in the format where there are two characters then the @ symbol, at least two more characters, a . , then finally at least two more characters

Views: a form of a table which is made of other columns from one or more pre-existing tables or views in the database

**References**

"RateMyProfessors.com – Find and Rate Your Professor or Campus." Rate My Professors. Web. 3 Apr. 2015. <http://www.ratemyprofessors.com/>.

*Central Intelligence Agency*. Central Intelligence Agency, n.d. Web. <https://www.cia.gov/library/publications/the-world-factbook/>

*Wikipedia*. Wikimedia Foundation, n.d. Web. 21 May 2015. <https://www.wikipedia.org/>

Game of Thrones TV show and Book series by George R. R. Martin

Example final report on the course website

**INDEX**