Phase 3 Report

Queries

The first query is a query that selects all theories from the current user and displays the theory ID, the theory description and the theory status. This query allows the user to see what the theories are and which theories are right and which theories are wrong. The parameter taken from the user is their respective user ID which is automatically determined when the user logs into their account.

The second query is a query that gets the user's accuracy based on their theories. This query takes in the parameter of the current user ID and displays the percentage of correct and incorrect theories that they have submit to the database. This displays the accuracy of the current user's theories based on their validity.

The third query allows the users to see some of the top theory creators. This query highlights three users who have created more than three correct queries. This is to highlight users that are active users and encourage them to use the application, a bit like a leader board. There is not a parameter provided by the user.

The fourth query allows the users to set the status of the current user's theory. Any theory created by the user can be set to either in progress, correct or incorrect. When a user creates a theory, the progress of their theory is set to in progress. Once the theory has been proven correct or incorrect the user can set their theory status to correct or incorrect. The parameters provided by the user are the theory ID and the new changed status.

The fifth query returns the character information based on the one character. The user will select a character and information regarding the character profile will be returned including their name, their title, their status and their religion. The parameter provided by the user is the name of the character.

The sixth query returns all relationships relating to a single character. The user will select a character and all relationships involving that character will be returned. The provided by the user is the character name.

The seventh query returns the house information relating to a single house. The user will select a house name and the information about the house will be returned including the house name, the house banner, the house saying and the house status. The parameter provided by the user is the house name.

The eighth query returns all of the characters that belong to a house. The user will select a house and the members of that house will be returned. The character information including the character ID, their name, their title, their status and their religion will be returned for all characters in that house. The parameter provided by the user is the house name.

The ninth query returns the houses that belong to a faction. The user will select a faction leader and the characters that follow that faction will be returned. The house name, the faction

name, the faction status and the leader name are returned for all characters in that faction. The parameter provided by the user is the leader name.

The tenth query adds a character to the database. Any user can add a character to the database. When a new character appears in the series the new character will be added to the database. The character is added by the user to the database when the user provides the name, title, status and religion. The parameters for this query are the character name, the character title, the character status and the character religion.

The eleventh query returns a list of the living characters. The user will query the living characters and a list of the living characters will be returned. There is no parameter for this query rather this will load along with the page.

The twelfth query updates the status of a character when the character has been killed. When a character has been killed the status of the character must be updated to dead. The user will input the name of the character and their status will be updated to dead. The parameter provided by the user is the character name. The only way to delete a user is from the writer page.

Index

The team decided to index on the table *mainCharacter* on *charID* because it is the largest table that can handle an index. The team decided it would be most effective on the largest table. Through testing of the queries with and without the index it was determined that the tables do not have enough data to actually be effective. There are only 2046 rows in the *mainCharacter* table and the index does not change the time it takes to run the queries in the database. The team tested this by timing the queries with and without the index. This is good for the database as it is always efficient but the hope is that if additional rows will be affected by the index.

Changes

There were two tables that were converted into one table. The previous reader and writer tables were converted into one table called *mainUser* because the functionality of the application made more sense when all users have the ability to update the database. In the application there is a reader page and a writer page and the database can only be updated from the writer page. It is incorrect to assume that the writer will always update the database because that means the database is reliant on the writer to keep it up to date. Combining the reader and writer tables and allowing users to make changes to the database will ensure that the application is up to date with the current series.

In addition the character ID will now auto-incremented when the table is created so that characters can easily be added to the database. The data in the *HasATheory* table was changed as well. The team added a theory status to this table so that users could determine whether or not the theory was in progress, valid or invalid. This is an important part for the functionality of our application.

As requested, our team made a few changes to our queries. An aggregate GROUP BY query was added to our list of queries and the query that did not work was changed to a query that did work. A majority of the queries were changed. Once the team started implementing the application it was determined that the previous queries were not a perfect implementation of the application and it made more sense to implement queries with data that users would want to see. The required query types are still included in the database they are just not the exact same queries that were submitted in phase two.