Tristan Santiago
Group 55
John Casillas
CS 340_400
Project Step 5 Final Version: ERD, Schema & DDQ

Database URL: http://flip3.engr.oregonstate.edu:42054/

Step 1: Peer Review Feedback

This sounds like an awesome, fun, and potentially actually useful project! This is all really well thought out. Looks like you have everything you need.

- Stephanie Gritz, Jul 6 at 4:57pm

This looks like a very interesting project. The outline looks good and is easy to follow. I do agree that the new design is more complicated than the first. Personally, I would switch to the original design just so every table is more simple and easy to follow. If you are up for the challenge, the new one has a lot of potential, but could also end up messy without careful planning. Hope this helps.

- Ryan Sisco, Jul 6 at 8:56pm

---*We decided to overhaul our project. This wasn't due to feedback, but rather after attempting to create the schema and ERD, we ran into a plethora of issues. We wanted to create something simpler, but more detailed.*

Step 2: Peer Review Feedback

"This looks like a really interesting project. - You may want to **add some detail to the outline that describes how a user would interact with your database.** What type of queries could they run, etc.. - I'm curious how you will limit the possible entries for the server attribute. I have a similar problem and I found check statements don't prevent bad entries. I'm using an ENUM for now, but if you come up with a good solution I would love to hear it! - playerKills and playerDeaths could be in the Player entity and brought into the playerTeam relation with a query. Similarly teamKills and teamDeaths could come from the Team entity. - champID says it will be assigned to each team as it is entered. Is this specific to a character or a team? - I'm not sure how you could limit the winning team entry to be one of the teams in that game, but it would be cool if you could. - I'm not clear on what champBan means since I don't play the game. **You may want to give some additional description in your outline for people who are unfamiliar.** The schema and ERD both look great. They are clear and seem to match the design perfectly. The sql file also looks good and matches the design exactly with all foreign keys correctly identified. Looks like a fun project. Good luck!"

- Justin Clarke

“Very cool, especially since I've logged many hours on the Rift myself! I'm pretty impressed how you guys were able to take everything that is tracked in the game (...a lot of stuff) and break them down into their entities and respective attributes. The ER Diagram and Schema seem to have everything you need. Great how you used different color arrows in the schema for easier readability.”

- Christopher Ragasa

TA Feedback

“Suggestions:
For simplicity we can merge entities that have one to one relationship if one of them don't have other relationships with other entities.

playerTeam looks like a recursive many to many relationship among players. It is an intermediary table that build this relationship and has its own attributes (relationships can have attributes too)”

- Abtin Khodadadi

Actions Taken Based on Feedback

- We agreed with Justin’s feedback and added some detail to the outline describing how a user would interact with the database. We also included some additional descriptions to make things a little clearer for individuals that don’t play the game.
- Our TA suggested we merge entities that have a one-to-one relationship if they don’t have relationships with other entities. This caused us to reflect on what the champBan and champPlay relational tables were doing, ultimately remove them, and simply setup foreign keys for champBanID and champPlayID under the playerTeam relational table.

Upgrades to the Draft Version:

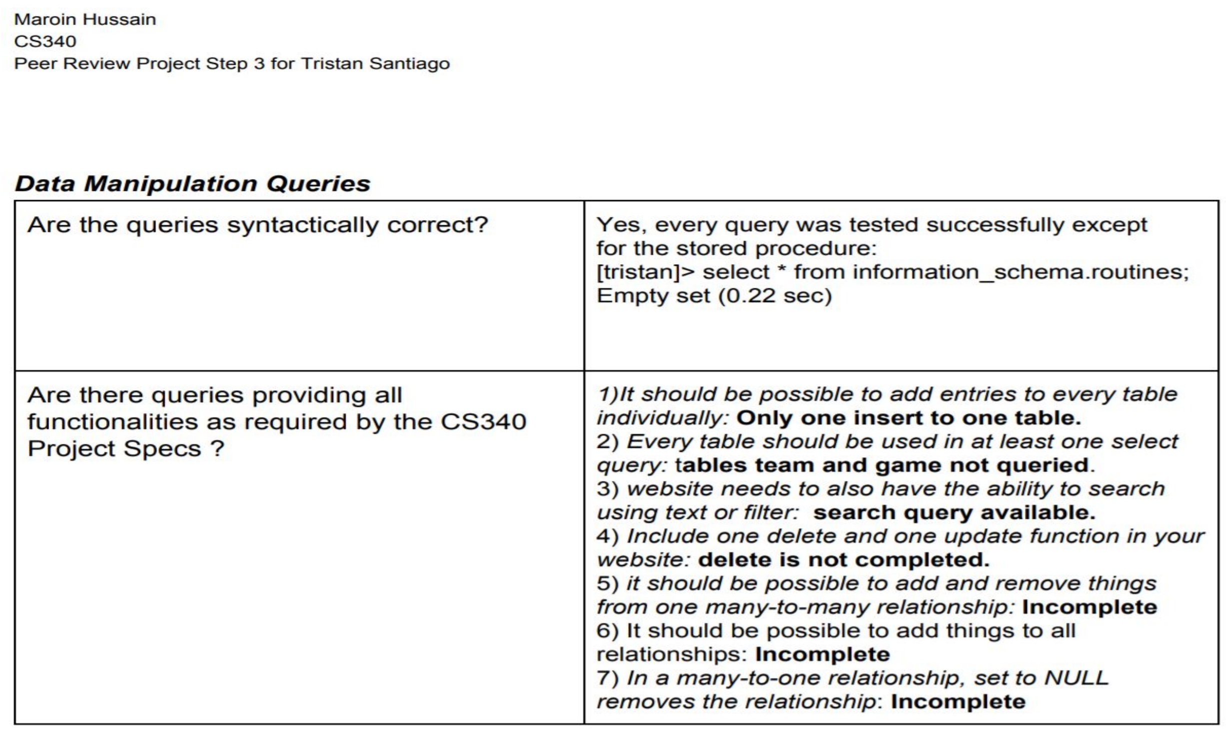
We have made the following changes to the draft version:

- Implement sample queries to serve as an example of how one might interact with the database.
- Included additional detail for individuals that do not play the game.

Step 3: Peer Review Feedback

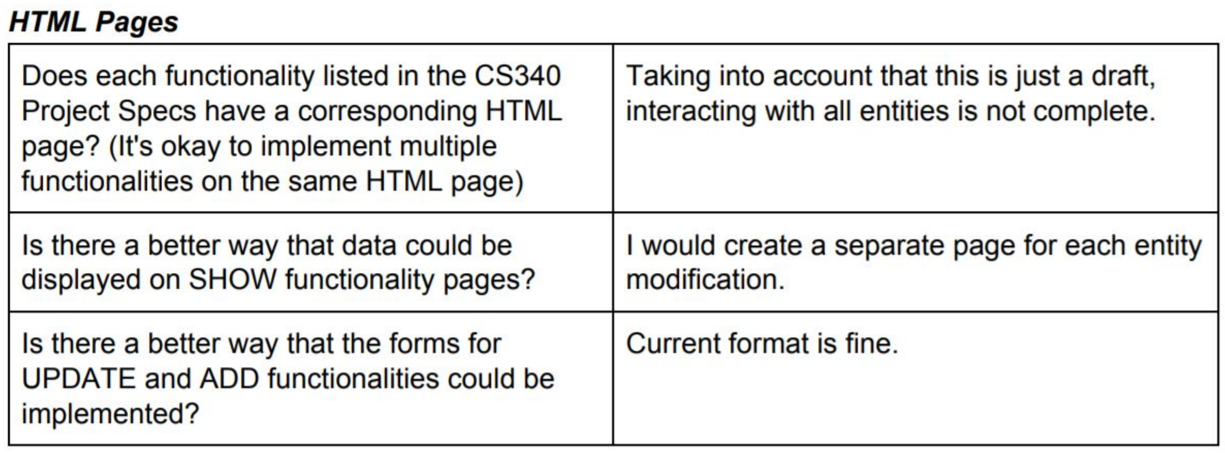
“Hi Tristan and John, here are some of my thoughts after looking at your step 3 draft. The queries you have all look syntactically correct, however I think you may need to add a few queries to your list. The project specs mention that you should be able to add entries to every table individually, but it currently appears that you are missing queries to add to the **playerTeam, champ,** team, and game entity tables**.** Your HTML page that you linked to looks really nice and is a great start. I curiously tried adding a new player and was surprised when it actually worked! That was cool to see, you definitely have a head start on that. The update and delete functionality looks covered as well. I think you'll need to add a few things to the page still, based on the project specs. It would be neat to be able to look up a certain players team mates, or look at a **player's won/lose ratio.** I would add a search or drop down menu for the players in the database so that you could bring up a certain player's stats. What you have currently looks great though, so if you keep that up your end result will be awesome. Cool idea and good job!"

- Wesley Cwiklo,


Maroin Hussain
CS340
Peer Review Project Step 3 for Tristan Santiago

Data Manipulation Queries

Are the queries syntactically correct?	Yes, every query was tested successfully except for the stored procedure: [tristan]> select * from information_schema.routines; Empty set (0.22 sec)
Are there queries providing all functionalities as required by the CS340 Project Specs ?	*1)It should be possible to add entries to every table individually:* **Only one insert to one table.** 2) *Every table should be used in at least one select query:* t**ables team and game not queried**. 3) *website needs to also have the ability to search using text or filter:* **search query available.** 4) *Include one delete and one update function in your website:* **delete is not completed.** 5) *it should be possible to add and remove things from one many-to-many relationship:* **Incomplete** 6) It should be possible to add things to all relationships: **Incomplete** 7) *In a many-to-one relationship, set to NULL removes the relationship*: **Incomplete**


HTML Pages

Does each functionality listed in the CS340 Project Specs have a corresponding HTML page? (It's okay to implement multiple functionalities on the same HTML page)	Taking into account that this is just a draft, interacting with all entities is not complete.
Is there a better way that data could be displayed on SHOW functionality pages?	I would create a separate page for each entity modification.
Is there a better way that the forms for UPDATE and ADD functionalities could be implemented?	Current format is fine.

TA Feedback

“Suggestions:
For simplicity we can merge entities that have one to one relationship if one of them don't have other relationships with other entities.

playerTeam looks like a recursive many to many relationship among players. It is an intermediary table that build this relationship and has its own attributes (relationships can have attributes too)”

- Abtin Khodadadi

Actions Taken Based on Feedback

- Wesley mentioned that we needed to add a few more queries to our list, but we weren't exactly sure which queries he might have been thinking of or referring to, as he did not explicitly state which. After careful consideration, we found that he was right and we were missing some queries, so we added team,and game queries.On that note, Maroin mentioned that our delete queries are incomplete. I believe this is a mistake as we our utilizing a stored procedure that handles our deletes for us.
- Wesley also pointed out that the site should allow users to add entries to every table individually. We did not have a chance to include those for the draft version of the outline, but we’ve made sure to those missing entities for the final submission.
- Wesley also suggested adding a drop down menu for players in the database so that one could bring up a certain player's stats. Although this is a feature we would like to add in the future, we weren't able to implement that functionality for this step. We instead decided to begin considering designing that feature to include later on in the project.
-
- Our TA suggested we merge entities that have a one-to-one relationship if they don’t have relationships with other entities. This caused us to reflect on what the champBan and champPlay relational tables were doing, ultimately remove them, and simply setup foreign keys for champBanID and champPlayID under the playerTeam relational table.

Upgrades to the Draft Version:

We have made the following changes to the draft version:

- Implemented sample queries to serve as an example of how one might interact with the database.
- Included additional detail for individuals that do not play the game.
- Added tables for the entities specified in this outline.
- Added additional queries into the DMQ
- Added a wiki page for our champs to the html
- added several new features and overhauled our player edit system

Project Outline

Imagine you aspire to be a professional gamer, or just a better player overall, in your favorite game (League of Legends): You would want every tool available to you to give you an edge. What if you could query the entire database for your favorite video game to truly analyze that data? “How well have I been doing with this playable character, ie. ‘champ’?” “How well do I do when I play with my friend, Tristan?” Then you might think, I really need to keep adding my game stats to continue tracking my progress! This is the idea behind this project.

We will be making a database representing League of Legends gameplay. League of Legends is a multiplayer online battle arena (MOBA) video game developed and published by Riot Games. Teams of 5 battle one another on a map called Summoner’s Rift. League of Legends contains many different unique characters that can be played in a variety of ways in a number of different roles. The complexity of the game makes it an ideal candidate for a database.

Users of the database will be able to input gameplay statistics and use that data to deepen their understanding of their individual performance, which we hope will help them become better players overall.

<u>Database Outline in Words</u>
The entities in the database are:

- **player** [Entity 1] A representation of the actual user of the database.
    - **playerID**: This unique number is auto-incrementing and is automatically assigned to each player when they are inserted into the database.
    - **userID**: Name of the player which is a string of maximum 50 characters (arbitrarily chosen). It cannot be blank and there is no default.
    - **server**: This refers to the region on planet Earth where the player plays. Examples are NA(North America), Asia, SA, etc. Cannot be NULL and there is no default
    - **email**: This is where we keep the users’ email addresses. It can be queried upon later if necessary. Can be blank.

- **team** [Entity 1.5] This table is simple. It only holds the ID and team color. All other ‘team’ stats will be held in the playerTeam relation table.
    - **teamID**: This unique number is auto-incrementing and is automatically assigned to each team when they are inserted into the database.
    - **teamColor**: Color of team is boolean. Has to be either 1 = Red, 0 = Blue. It cannot be blank and there is no default.

- **playerTeam [**Entity 2] This table shows how each player from their individual team did in a given game. This is where the game’s stats will be held.
    - **playerTeamId**: This unique number is auto-incrementing and is automatically assigned to each team when they are inserted into the database.
    - **playerID**: Integer. This is pulled from the player table. Required and can’t be blank/null.

    - **teamID**: Integer. This is pulled from the team table. Required and can't be blank/null
    - **playerKills**: Integer, default = 0. Total number of kills per individual player.
    - **playerDeaths**: Integer, default = 0. Total number of individual player deaths.
    - **teamKills:** Integer, default = 0. Total number of combined player kills per team
    - **teamDeaths:** Integer, default = 0. Total number of combined player deaths per team
    - **barons:** Integer, default = 0. Total number of Baron kills per team.
    - **towers:** Integer, default = 0. Total number of Towers destroyed per team.
    - **champBanID**: Int, foreign key of the champ table. This shows which player banned which champion.
    - **champPlayID**: Int, foreign key of the champ table. This shows which player played as which champion.

- **champ** [Entity 3] This table represents the individual playable characters in League of Legends.
    - **champID:** This unique number is auto-incrementing and is automatically assigned to each team when they are inserted into the database.
    - **champName**: This is the name given for the champion. String. NOT NULL
    - **atk**: This is the default attack damage this champion inflicts. Integer, default = 0.
    - **def**: This is the default defense this champion has. Integer, default = 0.

- **game** [Entity 4] This table houses the playerTeam data to reflect who won or lost a game.
    - **gameID**: This unique number is auto-incrementing and is automatically assigned to each team when they are inserted into the database.
    - **team1ID:** This integer is from the playerTeam table for the first team that competed this game. This has to be present, as it tracks who won or lost.
    - **team2ID:** This integer is from the playerTeam table for the second team that competed this game. This has to be present, as it tracks who won or lost.
    - **winningTeamID:** This is one of the two numbers from above.

The relationships in our database are**:**

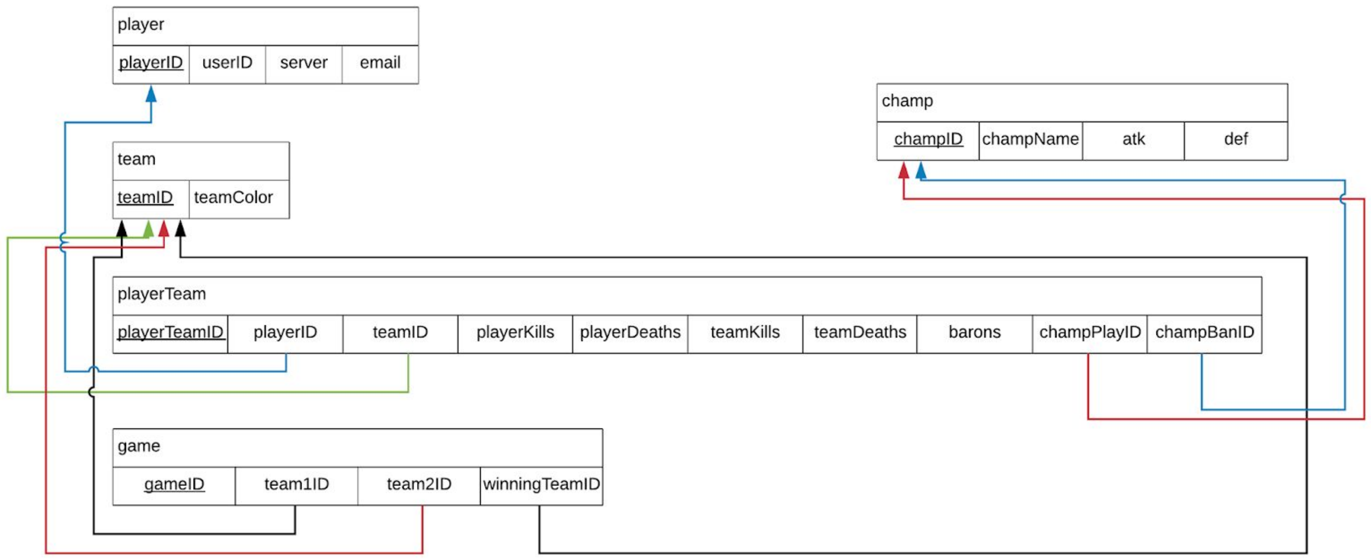
- A human creates a 'player'. This player is a part of a team. A player must be on one team, but a team has to have several players. This is a one-to-many relationship.
    - We even created a special table to track the stats between teams/players.
- Every game has a winner or loser. - The game table houses who won or lost. It uses the playerTeam table as the relationship between the player and team tables. One team has to win, and one team has to lose. This is a one-to-one relationship.

- A team of players has to play the game, and a game can only be played by a team of players. This is a <u>many to many relationship.</u>

<u>Post-Concept Thoughts</u>

In our second attempt at this outline, we have re-done just about everything. We created more tables to house relationships between the primary entities, as well as considerably simplified our end goal. Previously we were going to track the same data, plus in individual rosters. It was going to be a fantasy league. The issue we ran into were all of the circular joins that would've been necessary. This is a much simpler and linear database for our first attempt.

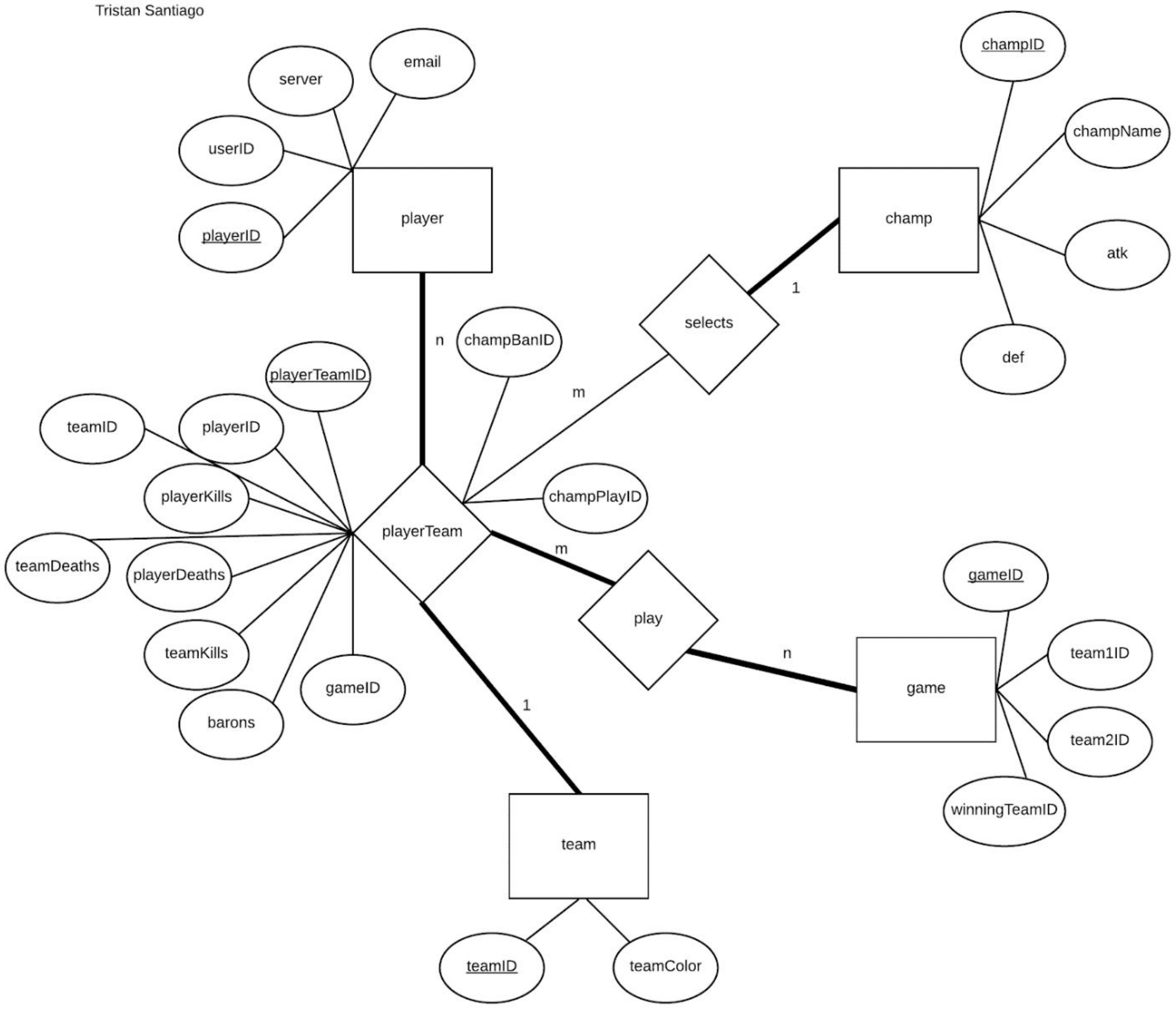
SCROLL DOWN FOR ERD AND SCHEMA!!!



Link to a higher quality version of the image posted above:
https://www.lucidchart.com/invitations/accept/f7e97a41-c96a-46f6-8cc5-f739593f4341

Project Step 2 Draft Version: ERD, Schema & DDQ

John Casillas
Tristan Santiago



https://www.lucidchart.com/invitations/accept/d2b37b6c-4eaa-4749-b031-3c90223355b8
Please click this link for better quality of the above.