Colin Van Overschelde
CS 340
6/12/2018
Project Step 8: Final Project

Overview

My project will be to build a database that contains data representing Players, Teams, Matches and Leagues for the popular video game Overwatch. Overwatch is a team-based, online, first-person shooter where teams of 6 players compete against each other to attack, defend, and complete objectives. It is currently one of the largest professional gaming leagues, with gameplay built around nuanced strategy and coordination. But the only game mode available in the retail version, matches players somewhat randomly, in one-time matches. These spontaneous matches don't provide players with the environment necessary to develop teamwork or even decide who is playing what role/position, which results in a random, and quite frustrating experience. As will be demonstrated by this project, a database that provides players with the tools to form teams, join leagues and compete in scheduled matches is an easy solution to the issues players are facing.

You can visit the website for my project at: http://flip1.engr.oregonstate.edu:54862/

Fixes Based on Feedback

Feedback from Project step 1: None

Feedback from Project step 2:

From Anonymous User: "*The ERD did not show any attributes other than the various primary keys. The relationships matched, however, the cardinality was not consistent between match and league. The outline says it is a M:N relationship under league, 1:1 under match, and the ERD shows it as M:1. Regarding the schema, I found it difficult to follow because the schema after reading the description and reviewing the schema because there are two relational entities listed but the ERD led me to believe that there might be four. Also the outline seemed more technical than necessary. Plain language would make it easier to follow the broad stroke concept for people who aren't into video games. Also, the match listed tie as a boolean when I believe that it would be a bit field serving as a boolean. I understand what the endstate is I think but things like that just make it more challenging to grasp the concept. Also, in the scheme m_time is shown twice. Otherwise, the entities in the schema and outline match*"

Fixes: All attributes for each entity have been added to the ER diagram in this version. I have also updated the cardinality to properly reflect the relationships. I remove the duplicate entry of m_time in the match entity of the schema. The feedback for the tie attribute of the match entity was corrected by creating a new table called result that is referenced by the match table. This will allow the Match to hold a variety of values that more accurately describe the current state of the Match Result. I did my best to better describe the entities in more plain language to make the outline more readable for people who aren't familiar with this particular video game.

From Tanya Khemani:

"*Mention the participation between relationships as you learned in Week3 and you need to specify at least four relationships*"

I have updated my outline, schema and ERD to explicitly call out the participation between each relationship and have created more than 4 relationships to demonstrate 1 to many and many to many relationships.

Feedback from Project Step 3:

From Tanya Khemani:

"*The SQL file gives error after importing it to PhpMyAdmin*

The data types look different from the outline

(-2) - Foreign keys are defined correctly except t_name

The relationship tables are present"

Fixes: I updated the order of my table creations to create tables in the proper order so that all foreign keys are available. I also added the appropriate tick marks (`) to the `match` table so that the protected keyword is allowed. I have confirmed all data types with my outline. I was unable to find the foreign key issue with t_name, as t_name is not being used a foreign key in any table.

Feedback from Project Step 4:

From Anonymous User:

"*HTML*

1. All functionality have a corresponding HTML page.

2. Show functionalities were clear and can't recommend any changes.

3. Functionality to join a league was not very clear. Maybe have a layout that is more obvious when user has joined a team other than the leave/join button change.

Data Manipulation Queries

1. Queries were all syntactically correct.

2. There is a query to add each entity (team, league, player, match, results)

3. Relation queries are correct.

4. Yes update satisfies rubric row 20.

5. There is a query to update many to many relationship (team-league)

6. there are correct queries for deleting many to many relationships.

7. There are queries to delete entities.

8. There is a filter functionality with league availability."

Fixes: To make the functionality more clear, I have updated many of my HTML pages to provide a better interface for demonstrating the database operations. I have also updated my Data Manipulations Queries file to include many additional queries that were required to display the data on the live website.

Feedback from Tanya Khemani: "*Looks Good*"

Fixes: No changes

Feedback from Project Step 5:

"*Feedback from previous steps missing.*

-15: I am not able to access your database, if you can justify it, I can re-grade it"

In my previous submission, although I included the changes I made, I failed to include the actual feedback from my peer reviews and graders that prompted me to make the changes. In this version I have been sure to include the feedback from each step, along the with action taken based on the feedback.

Feedback from Project Step 6: None

Feedback from Project Step 7: None

Latest Revision

After further evaluation, I made major revisions to my website design. The presentation of my last submission met most of the requirements, but didn't have a large enough set of data to properly communicate meaning from the data. For this revision, I updated my pages to be more of an administrator view rather than a customer facing view. This allowed me to simplify the presentation and provide additional functionality to better demonstrate the contents and manipulation of my database. In this version, I have updated the Players page to display the full roster for each Team. I also added a dropdown selector that allows the user to select a Team to filter by, and the page will update to display only the Players that are on the selected Team. I have also added functionality to fully edit all aspects of the Match entity.

Database Outline

Project Description
My project is to create a database that represents competitive leagues for the popular video game Overwatch. In this game, each Match pits 2 Teams of 6 Players against each other in a head to head first person shooter competition. My database will describe and maintain records of the Players, Teams, Leagues and Matches that place.

For more information on the game, visit playoverwatch.com and for more information on the professional league, you can visit overwatchleague.com.

The database will consist of the following entities:

- Player
- Team
- Match

- League
- Result

Player

The Player entity will represent an individual user of the video game Overwatch. A Player participates on a Team of 6 human Players, and competes head to head against another team of 6 human Players. Players will contain attributes which describe their in-game profile and can be related to many Team entities. There were many attributes that were removed from the original design because the Player entity wasn't the source of this data. Instead of including these attributes in the Player entity, they will be referenced through relationships within our database and external lookups.

Player Attributes

- p_id - Primary key to identify and reference a player
  - Data type: INT(11)
  - Constraints: NOT_NULL, Primary Key
- p_tag - Players battletag, which identifies the player with the server
  - Data type: VARCHAR(255)
  - Constraints: NOT_NULL
- p_name - Display name for the player
  - Data type: VARCHAR(255)
  - Constraints: n/a

Player Attributes Removed from design:

- p_level - In game player level
  - Data type: INT(11)
- p_skill_rank - In game player skill rank
  - Data type: INT(4)
- p_wins - Win count for the player
  - Data type: INT(6)
- p_losses - Loss count for the player
  - Data type: INT(6)
- p_ties - Tie count for the player
  - Data type: INT(6)

Player Relationships

- Player-Team
  - The Player-Team relationship will be a many to many relationship managed through the Player-Team table
  - A Player can exist without being related to a Team, likewise, a Team can exist without being related to any Players

Team

The Team entity will represent a team of Players. Team attributes describe the team which is nearly identical in concept to a regular sports team. Several attributes have been removed from the original design of the Team table. These were removed because they were not the source of the data. When a Team needs these values, the query will be joined with the appropriate table.

Team Attributes

- t_id - Primary key to identify and reference a Team
  - Data Type: INT
  - Constraints: NOT_NULL, Primary Key
- t_name - Display name for the player
  - Data Type: VARCHAR(255)
  - Constraints: Unique

Team Attributes Removed from design

- t_skill_rank - In game player skill rank
  - Data Type: INT(4)
- t_wins - Win count for the player
  - Data Type: INT(6)
- t_losses INT Loss count for the player
  - Data Type: INT(6)
- t_ties INT Tie count for the player
  - Data Type: INT(6)

Team Relationships

- Player-Team
  - The Player-Team relationship will be a many to many relationship managed through the Player-Team table
  - A Player can exist without being related to a Team, likewise, a Team can exist without being related to any Players
- Team-League
  - The Team-League relationship will be a many to many relationship managed through the Team-League table
  - A Team can exist without being related to a League, likewise, a League can exist without being related to any Teams
- Team-Match
  - The Team-Match relationship will be a many to many relationship. Since this relationship limits each Match to be related to 2 and only 2 Teams, it will be managed through the Match table
    - A Team can exist without being related to a Match, however, a Match must be related to two Teams

Match

The Match entity will represent a scheduled match between two opposing teams, and conceptually is no different than a regular sports match/game. A match is scheduled at a specific day and time, within a certain league, between two opposing teamsThe m_day attribute was added to the original design so that the m_time attribute was reduced to the smallest possible description. The m_winner and m_loser attributes were consolidated to m_result, which will allow the Match to contain one of many different Result options. This will require a new relationship Match-Result that wasn't included in the original design.

Match Attributes

- m_id - Primary key to identify and reference a Match
  - Data Type: INT(11)
  - NOT_NULL, Primary Key

- m_time - Time of the Match
  - Data Type: TIME
  - Constraints: NOT_NULL
- m_day - The date of the Match
  - Data Type: DATE
  - Constraints: NOT_NULL
- m_league - l_id of the related League
  - Data Type: INT(11)
  - Constraints: NOT_NULL
- m_home - t_id of the Home Team
  - Data Type: INT(11)
  - Constraints: NOT_NULL
- m_away - t_id of the Away team
  - Data Type: INT(11)
  - Constraints: NOT_NULL
- m_result - r_id of the Match Result
  - Data Type: INT(1)
  - Constraints: NOT_NULL

Match Relationships

- Match-League
  - The Match-League relationship is a many to one relationship managed by Match table containing an attribute, m_league, which contains the l_id of the related League
  - A Match must be related to 1 and only 1 League, however a League can exist without being related to any Matches
- Team-Match
  - The Team-Match relationship is a many to many relationship managed by the Match table, which contains the m_home, m_away, m_winner, and m_loser attributes, which contain the t_id values of the related team.
  - A Match must be related to 2 Teams, however, Teams can exist without being related to any Matches
- Match-Result
  - The Match-Result relationship is a many to one relationship managed by the Match table, which contains the m_result attribute, that holds the r_id value of the related Result
  - A Match must be related to 1 and only 1 Result, however a Result can exist without being related to any Match

League

The League entity will represent a League, which is conceptually no different than a recreational sports league. The l_division attribute was added to the original design to add the ability to provide divisions within each skill tier.

League Attributes

- l_id - Primary key to identify and reference a League
  - Data Type: INT(11)
  - Constraints: NOT_NULL, Primary Key

- l_day - Reference to the day of the week this league plays its Matches. Values from 0- 6, corresponding to each day of the week
  - Data Type: INT(1)
- l_skill_rank - The rank tier for the league. Values from 0-6 represent the 7 skill tiers of the retail game mode
  - Data Type: INT(1)
  - Constraints: l_skill_rank, l_division pair must be unique
- l_division - Values representing the division within the given l_skill_rank
  - Data Type: INT(2)
  - Constraints: l_skill_rank, l_division pair must be unique
- l_start_date - Date the League started
  - Data Type: DATE
  - Constraints: NOT_NULL
- l_end_date - Date the League ends
  - Data Type: DATE
  - Constraints: NOT_NULL

League Relationships

- Team-League
  - The Team-League relationship will be a many to many relationship managed through the Team-League table
  - A Team can exist without being related to any Leagues, likewise, a League can exist without being related to any Teams
- Match-League
  - The Match-League relationship is a many to one relationship managed by Match table containing an attribute, m_league, which contains the l_id of the related League
  - A League can exist without being related to any Matches, however, a Match must be related to 1 and only 1 League

Result

The Result entity will represent the result of a Match, providing additional descriptive data for a Match.

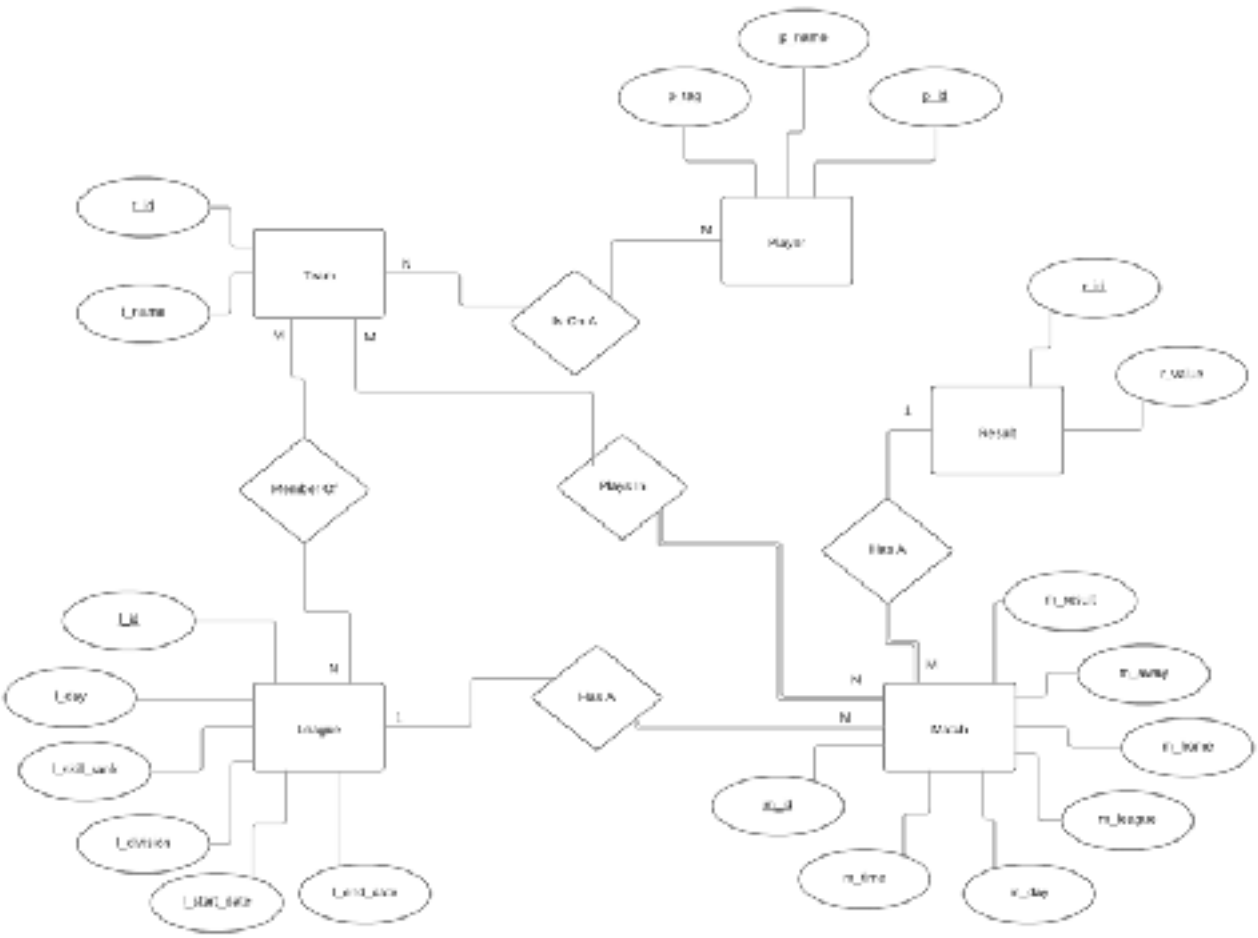
Result Attributes

- r_id - Primary Key to identify and reference a Result
  - Data Type: INT(11)
  - Constraint: NOT_NULL, Primary Key
- r_value - String containing the outcome/status of the Match
  - Data Type: VARCHAR(255)
  - Constraint: NOT_NULL, UNIQUE

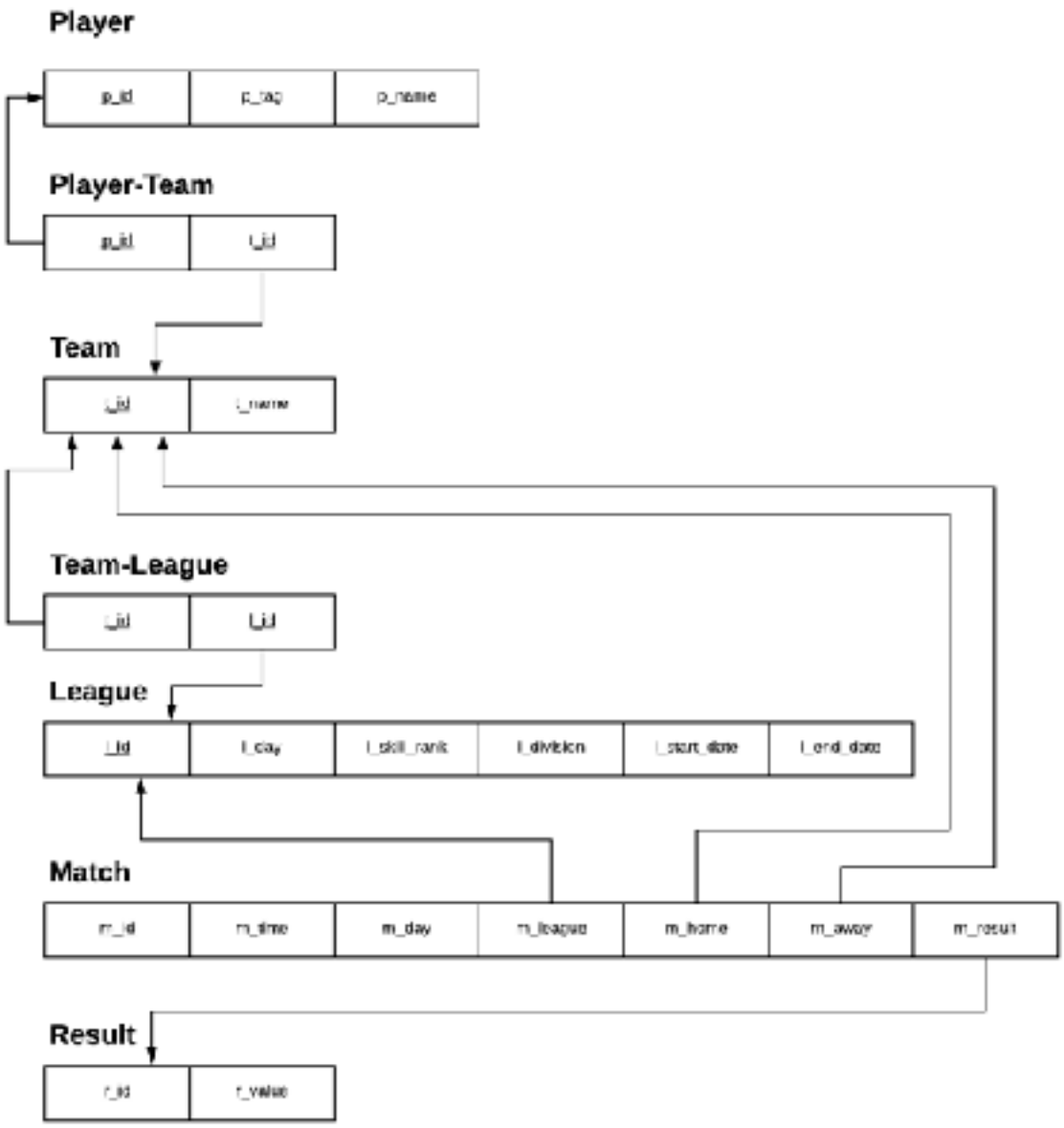
Result Relationships

- Match-Result
  - The Match-Result relationship is a many to one relationship managed by the Match table, which contains the m_result attribute, that holds the r_id value of the related Result
  - A Result can exist without being related to any Matches, however a Match must be related to 1 and only 1 Result

Entity Relationship Diagram



Schema


Player
p_id
p_tag
p_name
Player-Team
p_id
t_id
Team
t_id
t_name
Team-League
t_id
l_id
League
l_id
l_city
l_skill_rank
l_division
l_start_date
l_end_date
Match
m_id
m_time
m_day
m_league
m_home
m_away
m_result
Result
r_id
r_value