CS 2300 Database Project
Phase III

Jack Kufa

December 11, 2020

Problem Statement

I am building a Discord bot paired with a simple website, made to manage and automate a Pokemon Draft League. A Pokemon Draft League is a custom game mode for Pokemon battling, where coaches form teams and draft unique Pokemon to compete head to head. Creating an application to automate the internal work involved in running such a league would be extremely helpful in streamlining specific aspects of league upkeep such as updating rankings, tracking player wins, etc. The reason I am using a Discord bot to interface with the database is because Discord is a vital platform for communication between players in this system, and being able to query data in the same application as everything else related to the league would be very convenient. With that said, a simple website would be helpful for visualizing certain sets of information, such as the League's schedule and list of Pokemon that can be drafted or have already been drafted.

Revised ER Model

The five entity sets are as follows:

1. League: Primary entity that specifies league rules and battle format (single or double battle).
2. Users: Users who are participating in the league. The subclasses are as follows:
   (a) Coach: Think of it like the coach of a sports team. These are the players that are participating in the league
   (b) Administrator: A person who has additional privileges for managing the league.

   These subclasses are overlapping, meaning an Administrator can also be a coach.
3. Teams: Consist of a coach and Pokemon, and participate in matches. This is the equivalent of a sports team.
4. Pokemon: Entities who are listed on the draft list, sorted by tier/point value, and are chosen by teams for battling. These are like the players for a sports team.
5. Matches: contains information related to the games being played between 2 teams.

Logical Database Design

Summary of Data Types

Table	**Attribute**	**Type**	**Constraint**
League	Name	CHAR(50)	Primary Key
League	Format	BOOLEAN	
League_Rules	Name	CHAR(20)	Foreign Key
League_Rules	Rules	CHAR(20)	Multivalued
User	Discord_Username	CHAR(32)	Primary Key
User	Timezone	CHAR(5)	
Coach	Discord_Username	CHAR(32)	Foreign Key
Coach	Showdown_Username	CHAR(18)	
Administrator	Discord_Username	CHAR(32)	Foreign Key
Team	Name	CHAR(50)	Primary Key
Pokemon	Name	CHAR(20)	Primary Key
Pokemon	Serebii URL	CHAR(120)	
Pokemon	Value	INTEGER	
Match	ID	CHAR(25)	Primary Key
Match	Winner	CHAR(50)	
Match	Differential	INTEGER	
Schedules	League_Name	CHAR(50)	Foreign Key
Schedules	Match_ID	CHAR(25)	Foreign Key
Schedules	Week_No	INTEGER	
Team_Pokemon	Pokemon_Name	CHAR(20)	Foreign Key
Team_Pokemon	Team_Name	CHAR(50)	Foreign Key
Match_Team	Team_Name	CHAR(50)	Foreign Key
Match_Team	Match_ID	CHAR(25)	Foreign Key
Match_Players	Match_ID	CHAR(25)	Foreign Key
Match_Players	Winner	CHAR(50)	From players
Match_Players	Loser	CHAR(50)	From players
League_Users	League_Name	CHAR(20)	Foreign Key
League_Users	Users_Name	CHAR(32)	Foreign Key
Coach_Team	Coach_Username	CHAR(32)	Foreign Key
Coach_Team	Team_Name	CHAR(50)	Foreign Key

Functionality

The following is psuedocode for functions that will be implemented into the project. Please note that while the query syntax is made to mimic SQL, it is not 100% the same syntactically, and should not be treated as such.

- Basic Functions:

  1. Draft Pokemon:

```
!select <Pokemon Name>
  function SELECT_POKEMON(discord_username, user_message)
      if POKEMON contains user_message then
          pokemon = SELECT Name FROM POKEMON
                      WHERE Name = user_message
          coach = SELECT Name FROM COACH
                    WHERE Name = discord_username
          INSERT INTO POKEMON_COACH Pokemon_Name, Team_Name
          VALUES pokemon, coach
          print "You selected: " + pokemon
      else print  "ERROR. That is not a valid Pokemon!"
      end if
  end function
```

  2. Submit Replay:

```
!submit <Replay URL>
  function SUBMIT_REPLAY(user_message)
      if user_message starts with https://replay... then
          Parse website data for winner, loser, differential, and id
          INSERT INTO MATCH ID, Differential
          VALUES id, differential
          INSERT INTO MATCH_PLAYERS Winner, Loser
          VALUES winner, loser
      end if
  end function
```

  3. Replace (Modify) drafted Pokemon:

```
!redraft <Pokemon Name 1> <Pokemon Name 2>
  function REDRAFT_POKEMON($discord\_username$, $user\_message$)
      Split $user\_message$ into $find$ and $replace$
      if POKEMON contains $replace$ then
          $find\_pokemon$ = SELECT $Name$ FROM POKEMON
                            WHERE $Name = find$
          $replace\_pokemon$ = SELECT $Name$ FROM POKEMON
                               WHERE $Name = replace$
          UPDATE  $POKEMON\_COACH$
          SET  $Pokemon\_Name = replace$
          WHERE $Pokemon\_Name = find$
          print  $find$ + " Has been replaced with " + $replace$
      else print  "ERROR. That is not a valid Pokemon!"
      end if
  end function
```

4. Delete Pokemon:

```
!delete <Pokemon Name>
  function DELETE_POKEMON($discord\_username$, $user\_message$)
      if USER_ADMINISTRATOR.$contains$($discord\_username$) then
          DELETE FROM POKEMON WHERE Pokemon.$Name = user\_message$
      end if
  end function
```

- General Queries:

1. Query all of a user's usernames:

```
!userinfo <User>
  function USER_INFO($user\_message$)
      if USER contains $user\_message$ then
          R1 = INNER JOIN USER.$Discord\_Username$ = COACH.$Discord\_Username$
          $R2$ = INNER JOIN R1.$Discord\_Username$ = COACH_TEAM.$Coach\_Username$
          $info$ = SELECT $Discord\_Username, Team\_Name, Showdown\_Name$ FROM $R2$
                 WHERE $Discord\_Name = user\_message$
          print  $info$
      end if
  end function
```

2. Query differential:

```
internal function
  function DIFFERENTIAL($team\_name$)
    $positive$ = SELECT SUM $Differential$ FROM MATCH
               WHERE MATCH.$Winner = team\_name$
    $negative$ = SELECT SUM $Differential$ FROM MATCH
               WHERE MATCH.$Loser = team\_name$
    return $positve - negative$
  end function
```

3. Query wins:

```
internal function
  function WINS($team\_name$)
    $teamIDs$ = SELECT $Match\_ID$ FROM MATCH_TEAM
              WHERE $Team\_Name = team\_name$
    $wins$ = SELECT COUNT $Winner$ FROM MATCH_PLAYERS
            WHERE $ID = teamIDs$
    return $wins$
  end function
```

4. Query rankings:

```
!rankings
  function RANKINGS
    $rankings$ = SELECT $team$ FROM TEAM
               ORDERBY DIFFERENTIAL($team$) + WINS($teams$)
                         > DIFFERENTIAL($previous\_team$) + WINS($previous\_teams$)
    print $rankings$
  end function
```

5. Query matches played:

```
!matchesplayed
  function MATCHES_PLAYED
    $matches$ = SELECT $ID$ FROM MATCH
          WHERE $ID! = NULL$
    print $matches$ as URL
  end function
```

6. Query specific Pokemon:

```
!pokemon <Pokemon Name>
  function QUERY_POKEMON(user_message)
      if POKEMON contains user_message then
          pokemon = SELECT Name FROM POKEMON
                    WHERE Name = user_message
          print pokemon
      else print ”ERROR. That is not a valid Pokemon!”
      end if
  end function
```

7. Query Average Differential:

```
!average
  function AVERAGE
      SELECT AVG Differential FROM MATCH
  end function
```

This Functionality is tentative, and will change as the project evolves over time.