5330/7330 Fall 2025

Programming Homework

Due date: 10/31 (Wed) 11:59pm

For this homework you are to write a program that interacts with some database tables in MySQL.

Data – Bokamon players and matches

Bokamon is a very popular card game in the country of Timbuktu. There is a Bokamon association that keep track of all the players. It also wants to keep track of official matches that is being played.

For each player, we need to store his/her name, birthdate. his/her home state (which is represented by a two-character string).

Also, each player has an six-digit unique player ID assigned by the association. Furthermore, each player has a current rating (which is an integer >= 100).

Players play against each other in official matches. Each match is between two players. One of the two players will be denoted "the host", while the other is denoted "the guest". (The meaning of these terms is unimportant in this project, you just need to record who is what). For each match there will be a winner (no draws). You need to record the winner of the match. Also, after each match the ratings of both players may change. For each match we will need to record the old and the new rating for both players. Also, for each match we will record the start time (date and time) and end time of the match.

One thing we need to make sure is that no person can be involved in multiple matches at any given time. However, a player can play multiple matches on the same day (or same hour if the game is short enough).

Database tables

We will store the data in the following tables. (The detail schema (e.g. domain info) will be provided to you):

- Player – table store info about the team. Attributes include:
  - ID: 6-digit code that is unique for each player
  - Name: unique name for the team
  - Birthdate: Birth date of the player
  - Rating: Current rating of the player
  - State: name of the state that the team is based. We use the two-character abbreviations here (e.g. TX, NC, VA, YY)
- Matches – information about matches
  - HostID: player ID of "the host"
  - GuestID: player ID of "the guest"
  - Start: the date and time for the start of the match

  Notice that one can store a match that is yet to be played (or the result unknown). In such cases, all the following fields are set to NULL. In such case, the Start should be interpreted as a projected start date/time.

- End: the date and time for the end of the match
- Hostwin: Boolean, true if the host won, false if the guest won
- PreRatingHost, PreRatingGuest: Pre match rating of the host, and pre match rating for the guest
- PostRatingHost, PostRatingGuest: Post match rating of the host, and post-match rating for the guest. (Notice that the match may not have been played, if so these values are listed as null)

(Notice that you have the option to create a matchID as the primary key, or you can set your key such that the constraint can be maintained. You must add the constraints for this table).

Work to do

You are to write a program to manage the database tables. Your program should read in a csv file (if you use python, you can use the csvreader module to handle it easily). You should ask the user for the name of the file (do not hardcode it in your program). Each line of the file contains a command (in the first field), and a list of parameters (if needed), in subsequent fields.

The list of commands (the first field of each line) is as follows. Each command is a single character:

Data manipulation commands:

- 'e' : check if the tables exist. If not, create the tables.
- 'r' : clear all data (remove all tuples from all the tables, but do not remove the tables themselves), if the table does not exist then create them.
- 'p' : add a new player, the subsequent fields contain the following info in order
  - ID of the players (always 6 digits)
  - Name of player (can have space in middle)
  - Birthdate: format (yyyymmdd: example 20120327 corresponds to March 27th, 2012. this is the same for all date attributes below)
  - Rating: rating of the player (integer >= 100)
  - State (always 2 characters, accept any such string)
- 'm': Enter information about a COMPLETED match, the subsequent fields contains the following info in order
  - ID of the host
  - ID of the guest
  - Start time of the match: (in the following format: yyyymmdd:hh:mm:ss. For example, 20240209:18:37:45 corresponds to Feb 9th, 2024, 6:37:45pm) – Notice that time is in 24 hrs format)
  - End time of the match (same format as above)
  - Whether the host win: 1 is yes, 0 is false
  - Pre match rating of host
  - Post-match rating of host
  - Pre match rating of guest
  - Post-match rating of guest

- ‘n’: Enter information about a match TO BE PLAYED, the subsequent fields contains the following info in order
    - ID of the host
    - ID of the guest
    - Expected start time of the match: (in the following format: yyyymmdd:hh:mm:ss. For example, 20240209:18:37:45 corresponds to Feb 9th, 2024, 6:37:45pm) – Notice that time is in 24 hrs format)
- ‘c’ : enter the results of a game that is already in the database. The subsequent attributes are the same as those of ‘m’. If the game is NOT already in the database, you should REJECT the input.

Notes for all these commands, if the tuple cannot be inserted (due to violation of any constraints), you should print the whole line (as is listed in the input, and after that, print “Input Invalid” (in the same line).

You can assume that all the fields in the csv file is of the right format. (e.g., if a number is required, a number will be given)

Query and output commands:

- ‘P’ : return the information about a player. Subsequent fields contain.
    - Player ID

  The output should contain one line, that print (in order), ID, name, birthdate, current rating, state.

- ‘A’ : given a list of players, listed the win-loss against all other players that he/she has played against. Subsequent fields contain.
    - The playerID of all the players

  The output should be a list of lines, each line contains
    - ID of the player, name of the player, # of wins, # of lose, winning percentage (round to 4 decimal places)

  The output should be ordered by decreasing winning percentages, break ties with number of wins, and then playerID

  If the input contains only one player, output nothing. If the input contains the same player multiple times, treat them as one player. If the input contains a player that does not exist, ignore it.

- ‘D’: given a start date and end date, list all completed matches in chronological order of start time, break ties with the hostID. Subsequent fields contain:
    - Start date: format (yyyymmdd)
    - End date: format (yyyymmdd)

  The output should be the following:

- Each line should be one match, where the listing is start time of match, end time of match, name of host, name of guest, who wins (H for host, G for guest).

Extra query (bonus for 5330 students, required for 7330 students)

There is one extra query you can implement (bonus for 5330 students, required for 7330 students):

- 'M': return the matches by a certain player (regardless of whether it has been played), listed in chronological order. Subsequent fields contain.
  - Player ID

  The output should be the following.
  - First line: Player ID, Name of player
  - Each subsequent line should be a match, it should contain:
    - StartDate/Time, EndDate/Time, ID of the opponent, name of the opponent, whether the player win or lose (use W for win, and L for lost), post-match rating of the player.

  You should also note possible inconsistencies in terms of ratings. For instance, if the player's post-match rating for one game is not equal to the pre match rating of the next game, then you print "inconsistent rating" at the end of the line for the next game.

  Also, note that it is possible that a player has the result of one game, but result of an earlier game has not been entered, in that case it should not be flagged for inconsistent rating.

Other notes

In your program you should name use the user 'cs5330', with password, 'pw5330', and the database should be named 'dbprog'.

What to hand in

You are to upload the programs you implemented as a zip file. Maintain one-space between each field in every output line to maintain consistency.