Coursework 03

Deadline: May 11 **Weight: 25%**

Database for a social gaming network

This is a group project and your aim is to design and implement a database for a social gaming hub/portal in the style of Game Center on iOS. The database will support user activities, such as inviting friends to play a game, starting a multiplayer game through matchmaking, tracking their achievements, and comparing their high scores on a leader board. Imagine that you work on this project on behalf of a client (e.g. a company), who wants to setup the gaming network. Like many clients in this situation, the company probably don't know exactly what they want and there may well be problems with their specification which you will have to make sense of. They have given you a specification as shown below:

A. **Users**. These are complex objects. It is up to you whether you want to represent a user as one table or several, and you may decide to store some of the information elsewhere in the database.

   1. Users have a username, a first and last name, an email address, a password, a user-updatable "status" line, a creation date, the time of last login, and whether the account is currently logged in and online.

   2. They have a list of games they own. For each game, the user may give a rating (out of five), a comment, the date/time the user last played the game, the user's highest score and which achievements (see below) the user has gained in that game (and when they gained them).

   3. Users also have other stuff, such as an avatar (a url or pointer to an image), a friends list (pointers to other users) and a games-in-progress list (pointers to active games for which the user has asked to receive notification when that game is updated).

B. **Games**. Again, these are complex objects and it is likely that you will need to split off some of the information into separate tables

   1. Games will have a name, a text description, a release date, a publisher, a version number, a genre or category (ideally, you should allow for the game to be in more than one categories), an age rating, an average user rating (from all user ratings), an overall rank, a rank within its genre/category, a pointer to a default image, and an optional url for the game's website.

   2. Each game can have up to a maximum of 100 associated achievements (as described below).

   3. Each game has its own leaderboard. This shows the top 10 scores for that game.

C. **Achievements**. Each game has a number of achievements, which players earn by reaching a milestone or performing a particular action in the game. Each achievement has a unique ID, a title, a reference to the game it belongs to, a point value (there is a maximum of 100 points per achievement, and a total maximum of 1000 points for all the achievements belonging to any particular game), a flag to say whether it is hidden from the user until the user has earned it, two descriptions (one pre-earned, one for after the achievement is earned) and an icon (a url or pointer to an image).

What the client wants from you is:

A. An ER diagram and associated schema for the final database describing the various tables in the database and their keys and showing how the entities/attributes etc. are connected. Note that you can name and divide up the tables in whatever way you think is most sensible, and add ID numbers etc. as you think appropriate. **[20 marks]**

B. SQL statements to create the tables described in the schema. You should populate each of your tables with a small amount of dummy data. **[10 marks]**

C. SQL statements to perform the queries, triggers or procedures listed below. It is unlikely that you will be able to get all of these done – just do as many as you can. Some of these will require you to add new elements to your database: **[40 marks]**

    1. Given a game, list all the users who own that game

    2. Automatically update a game's average rating whenever a user adds or updates their rating for that game

    3. Change the database so that a game's average rating only appears if it has been rated by 10 or more users.

    4. Given a user and a game, display the user's score, rank on that game's leaderboard (even if they are outside the top 10) and an indication of where they appear relative to the average e.g. "85000 points - 1788 (Top 20%)" or "13000 points - 16364 (Bottom 40%)".

    5. Create a list of the top 10 rated games in each genre/category.

    6. To help prevent cheating, add an optional maximum and minimum score value to each game and check that user's scores are within this range.

    7. Add daily and weekly leaderboards for each game showing the best scores achieved this day or week.

    8. Check new usernames against a list of obscene or offensive terms (if possible check for sub-strings as well as the whole name). If such a username exists, lock the account.

    9. The client wants to add a 'Hot List' which shows the 10 games, which have been played most often in the past week – add the necessary fields, tables, queries, triggers and/or procedures to achieve this.

    10. The client wants to allow users to send friend requests to other users using their username or email address. What additional fields/tables would you need to add to allow for this functionality? You will need to keep track of the request, the response (if any) and some way of keeping track of user friends lists. If possible, extend this system to allow users to send an invite to a friend to play a particular game.

    11. Given a user and a game, show a leaderboard which lists just the user and their friends.

    12. List all of a user's friends and show whether they are currently logged on. If they are not logged on, show when they were last logged on and what they were playing.

13. Given a user and a game, show how many achievements they've got (out of how many in total the game has) and how many points worth of achievements they have for that game e.g. "16 of 80 achievements (95 points)"

14. Show a status screen for the player showing their username, status line, number of games owned, total number of achievement points and total number of friends.

15. Given a user and a game, list the achievements for that game (apart from ones which have the hidden flag set and the user hasn't earned). They should be listed with ones the player has earned listed first. For each achievement, list the title, points, description (which will vary depending on whether the player has earned that title or not) and when the achievement was earned.

16. Given a user and a named friend, for each game they both own, list the name of the game and the number of achievement points each of the users has for that game. If possible, at the end of that list add all the games that are owned by only one of the two users, showing how many points they have and a blank for the other user.

17. The client has realised that for some games (e.g. golf, driving games) lower scores are better. They want to add a "sort order" flag to the game to indicate this. They've also realised that many games have a format other than simply points when displaying scores. Add a "score format" field (e.g. int, time, money) to the Game table. The sort order and score format should be taken into account when displaying leaderboards.

18. Suggest another user as a friend, if they have a number of friends and/or games in common with the user. You should show the name of the potential friend and how many games/friends they have in common with the user.

19. Automatically suggest a game, which a user might be interested in. Games should only be suggested if the user doesn't already own them. Suggestions could be made on the basis that other users who make similar ratings also rate this item highly (though you may use another recommendation metric if you prefer).

20. Write another query/trigger/procedure of your choice, which really shows off what your database can do.

D. Write a report (8 to 12 pages) for your client, detailing what your system can do (showing actual output) and the decisions you have made (including clarifications and assumptions you have made about the specification). If you wish, the ER diagram and/or schema can be in an appendix to the report (so they don't count towards the page limit). You should list your references, including websites, and you should submit your database in a separate file (see below). **[30 marks]**

DBMS

It is recommend that you use MySQL for your project. It is freely available on the web and it should be straightforward to install on all major operating systems. You should submit a dump of your database, which we shall use during marking to test your database. If you decide to use Oracle, you do not need to submit a database dump; just tell us where to find your database and submit a .sql file with your SQL statements for table creation and the various queries. In any case, state clearly which DBMS you have used.

GROUPS

It is highly recommend that you work in groups of 5 people; you will find the project easier and you will get the most out of it with a group of that size. However, if you want to work in smaller groups or even work on your own then that's okay - your group size can be anywhere between 1 and 5 people.

You need to decide on groups yourselves and let me know via email at least a couple of weeks before the deadline, so I can enter them into the marking system: Dimitris.Vavoulis@bristol.ac.uk. Send me an email with the title "COMS20700 group statement" and attach a comma-separated (i.e. *.csv) file, where each line includes first the name of a group member, then his/her username and, finally, his/her relative contribution to the project (see below). The file should not include any other information (e.g. a header, comments, etc.). Submit a file even if you decide to work on your own. Each group should submit a single file.

MARKING

The final mark for each student will be based on the group mark adjusted by the individual relative contribution, as indicated in the text file submitted by the group. You, as a group, must agree the relative contribution for each member. For example, in a group of 4, the relative contributions should sum up to 1, e.g.:

```
Peter Parker, pp3094, 0.25
Bruce Banner, bb8746, 0.25
Tony Stark, ts4758, 0.20
Steve Rogers, sr3874, 0.30
```

In the previous example, Steve Rogers will receive 100% of the final mark, Tony Stark will receive 67% of the final mark and Peter Parker and Bruce Banner will receive 80% of the final mark each. Please make sure that every member of the group submits exactly the same final version of the coursework.

ACADEMIC INTEGRITY

All the work you hand in should be your own or your group's work. In general, any source other than the lectures should be explicitly cited at the point where it is used. If you are unclear about what to cite, you can ask me or see the department's guidance to plagiarism:

https://www.cs.bris.ac.uk/Tools/Local/Handbook/coursework.html

ABOUT GAME CENTER

More information on Game Center can be found in the links below. While it is not vital that you read all of this information in detail (you are building a system similar to Game Center, not trying to reproduce every aspect of it!), it may help to give some context and clarification.

http://en.wikipedia.org/wiki/Game_Center
https://developer.apple.com/game-center/

FINAL COMMENTS

This coursework is deliberately open-ended; the varying group sizes mean that different groups may come up with very different results. You are encouraged to use as many of the techniques you were taught during the course as possible; and research others, where required. It is unlikely that you'll be able to get through all of the tasks listed - just do as many as you can in the time available (this coursework should represent about 15 to 20 hours of work for each person in each group). In any case, make sure that you do at least part of each section of the coursework.

Good luck!