**CS-GY 6083 Project Part 1 (Minghao Shao & Yiming Li)**

**1.1. Relational Table Design**

1. User Table

The data started from the user table, which was used to store all the information may required about the user. For the primary key, a unique username was used, also used for login the session, instead of AUTO\_INCREMENT field. The username was designed and set by the user instead of generated by the system automatically each time a new account created. There were some benefits using this design, as a primary key, each time a user registering a new account, the system would check the uniqueness of this username, in which case the duplicate issue could be avoided since the system would check this, no manual action required. For the password field, instead of storing the password in plain text, SHA256 hashing function was used. In the test data, it was a simple SHA256 hashing, however, in the part2, more complex policy would be applied such as salting. Other information fields such as firstname, lastname, email, phone, city, state and country were designed to be nullable since sometimes user would prefer not to provide their personal information, hence this is not forced. Another field designed was the karma point. Which was used to identify the status of the user, regarding their level.

The policy regarding the karma point was according to the quantity and the quality of the answers the current user posted, say 10 points added each time the user posted an answer, and each time any answer the user posted received a “like”, the user could also get 10 points bonuses. Another facet about earning the karma point was when the user’s answer was selected by the question owner, the user who posted the best answer would also receive the bonuses, in test data, 20 points was set. Accordingly, like cancel was also taken into consideration, each time a user canceled the like, the 10 karma points bonuses would be dismissed also, when the best answer updated and the user’s the answer was no longer the best answer, the 20 points bonuses would also be dismissed. The mechanism of question or answer deletion was also designed and would be discussed in Questions and Answers tables section respectively.

1. Status

A separate table called Status was also design, which was used to store the information of each level of the user, at current stage, three levels were designed, basic, advanced and expert. In this table, statusid was used as primary key which was AUTO\_INCREMENT, also a status name, and the threshold called statuskarma, which stored the total karma point required for a user to reach this level. In this design, the basic level was from 0 to 50, advanced was from 50 to 100, and expert was larger then 100. Note that the numeric design was set only for the test data, to make some queries output the user with each level which could be optimized, for example, in part2, the threshold for each level would be increased, or the bonuses received by the user would be deducted.

1. UserStatus

The status of the user was not stored in Users table to satisfy the 3NF. Since the status of the user was totally depended on the karma points the user received, and the number of karma points received by the user was depended on the username. Hence, they were designed to divide to 2 separate table. The primary key was the username, which was enough to identity a user and the relation between the user and their status was 1 to many, which means a user could only have one status. From the statusid of each user, a join could be performed to get the exact status of the user from the status table.

1. GeneralTopics

The structure of the topic was designed as hierarchy. Two levels were designed and the GeneralTopics table was the first layer. In this table, only the general topic could be stored such as MATH, PHYSICS or COMPUTER SCIENCE. In this table, a AUTO\_INCREMENT field as primary key was design called gtid (general topic id), and another one was the name of the general topic.

The topics was aimed to be selected by a list box in the code implementation, there were several pre-defined topics, or the user may add topics on their own.

1. SubjectTopics

The next level of the topics was called SubjectTopics, in which the subjects were stored under their general topics. Similar as general topic, a stid (subject topic id) was design as the primary key, with AUTO\_INCREMENT, also with the stname which was the name of the subject topic. Each subject topic should be able to reference to their parent topic, hence a foreign key was set which refenced to the gtid of their parent topic. Such as a topic called ‘Database System’, which was certainly under the COMPUTER SCIENCE general topic, hence the gtid of this subject topic would be the gtid of the general topic CS.

Also same as the general topic, in code implementation they were designed to be selected with a list box after the general topic was selected when posting a new question. Some pre-defined topics would also be provided by the system. Or users were able to add new subject topics.

1. Questions

This table was designed to store the question information posted by the user. The primary key was called qid which was a AUTO\_INCREMENT field with int type. For each question, the username who posted the question was also recorded into the field q\_username. In consideration that users may either post questions, or give answers to questions, q\_username was used instead of the username, to make the table clearer, but the q\_username was still designed as a foreign key referenced to the username in Users table only the name of the field was different. In this design, general topic should not be used as the topic of the questions, but just to identity the subject topics which belonged to the general topic. Hence in the question table, a foreign key referenced to subject topic’s stid field was used to identify the topic of the question. Also, the title and q\_body field was used to store the title and the body of the question along with the post\_time. In the test data, the post\_time was inserted manually to show the test result and to simulate the real world condition, but the design for code implementation was always obtain the current time by the system by calling now().

The questions table also had a field called status, in this design, two statuses were defined, solved and unsolved, the user who posted the question could modify the status of the question. The status of the question would neither affect the karma points the user obtained nor affect how the question or the answers under this question would behave, but only a label which was used to make it convenient for the users who were also looking for the solution of similar questions. In in this, another situation was also considered for part2. Since in some cases, users would like to delete the questions they posted hence a field called a\_visible\_status was added to this table. Instead of deleting the question row directly, it would be safer to set the visible status of the question, if the question was not visible, then in the implementation in part2, it could also be regarded as deleted. Another benefit of using this field was that the user may be able to set the question as private/public in case they don’t want others to see these questions (in case some privacy issue involved).