12 May 2022

Online Forum Database System

Pengxiang Huang, Han Linghu, Zihan Li, Boyi Chen

School of Data Science, The Chinese University of Hong Kong, Shenzhen

Abstract--These remain the abstract

Index Terms—Remain the index term

—————————— ◆ ——————————

1. Background & Introduction

Our group observe that many student-programmers from CUHKSZ may encounter many familiar programming questions in their project or assignment. Their solutions mainly include: searching online, emailing TAs or professors, uploading questions in WeChat group, or assigning an office hour. Searching online sometimes may not be an efficient way, because the blog or some guidance information may not directly answer the assignment question. Even worse, programmers need to spend much time on filtering the huge amount of information and it becomes hard for them to get answers when the homework question is not relevant to the results on website. Uploading questions on WeChat group could get the detailed guidance and answer. But the new WeChat group will be created every semester for other students who may encounter the same problems. The connection between students already taken this course with the students taking this course right now is broken in this way. Raising questions in office hour is not convenient for programmers to solve their question immediately since they need to make an appointment and wait until that day comes. Therefore, our group would like to take the first step to change the current situation, to provide a online Q&A platform with the support of online forum database.

To make our database more efficient and space-saving, we made efforts to do normalization on tables and introduce index in our project. Additionally, we built fantastic UI in the frontend and robust backend server to hide the detailed implementation and manipulation of the database. So users can focus on the Q&A procedure, and they don't need to worry about how to get the desired information from the database, as all the queries are carried out by the functions in the frontend and backend.

1. Data Base Structure Design

*A. Requirement & Specification*

The main component of our project is the online forum database. There are three main entities in the database, namely “user”, “blog\_questions” and “blog\_answers”. The “user” entity stores the account information about the user, including their name, email address, password and etc., which is used for identity authentication when the user enters the online forum. The core of our project is that users can post a blog to raise a question and get relevant answers from other users, which brings about the “blog\_questions” entity and the “blog\_answers” entity to become the key features. Those two entities store the questions and answers information about the blog, including content, author\_id and etc. We also implement several relationship schemas to support the interaction between the user and the blog, for instance, user can like a blog or follow a blog. With the purpose of making our database become more efficient and space-saving, we spare efforts on doing normalization and applying index to our project.

*B. Entity-Relation Diagram*

The ER-diagram of our system contains 12 entities. The entities are user, group, sub\_group, blog\_questions, blog\_answers, attached\_file, attached\_picture, user\_like\_answer, user\_like\_question, user\_follow\_question, user\_follow\_group and user\_view\_question.

User Anyone who wants to have an access to our system needs an account, so the user entity has the related information of the user’s account, such as username, password, email address and the URL of profile photo. In the process of identity authentication, the system will throw an exception if it detects that the identity information does not match the records storing in the user entity.

Group In our system, each blog has a related group, such as CSC3170, which is convenient for users to find the blogs in the corresponding group. Moreover, it serves as the filter conditions in the process of searching, which improve the efficiency of searching. The group entity has group\_name, description, create\_time, update\_time, amount\_of\_follows. “Group\_name” is designed to store the course number of the computer science courses offered in CUHKSZ, such as CSC3170 and CSC4001. “Description” is designed to store the detailed information of the course, such as Database System and Software Engineering. “Create\_time” is designed to store the time period of the creation of this group. “Update\_time” is designed to store the time period of the update of the blogs belonging to this group. “Amount\_of\_follows” is designed to store the amounts of users who follow the group.

Sub\_group In our project, each blog not only has a related group, but also it has a related sub group, such as Assignment1. This can be served as the filter conditions in the process of searching, which improve the efficiency of searching. The sub\_group entity has group\_name and sub\_group\_name. “Group\_name” is designed to store the course number of the computer science courses offered in CUHKSZ, such as CSC3170 and CSC4001. “Sub\_group\_name” is designed to store the assignments in each course, such as Assignment1 and Project.

Blog\_questions The entity is of great significance in our online forum system. It has title, author\_id, group\_type, sub\_group\_type, content, code, like, follow, hot, create\_time, update\_time, views. “Title” is designed to store the title of the blog. “Author\_id” is designed to store the author of the blog, which is used to determine the authority of the deletion of the blog. “Group\_type” is designed to store the course number to which the blog belongs, such as CSC3170 and CSC4001. “Sub\_group\_type” is designed to store the corresponding assignment to which the blog belongs, such as Assignment1 and Project. “Content” is designed to store the content of the blog. “Code” is designed to store the code of the blog since our system provides an online code editor and complier. “Like” is designed to store the amounts of users who like the blog. “Follow” is designed to store the amounts of users who follow the blog. “Hot” is designed to store the popularity value of the blog, which is used to determine the order of the blogs displayed in “Hot Blogs” part. The value is highly related to likes, favors, views, and the creation time. The newer the questions posted, the larger the likes, favors, and views, and the higher the value of popularity. “Create\_time” is designed to store the time period of the creation of this blog. “Update\_time” is designed to store the time period of the update of the answers belonging to this blog. “Views” is designed to store how many times users have viewed the blog.

Blog\_answers The entity is of great significance in our online forum system. It has question\_id, father\_answer\_id, content, like, author\_id, create\_time. “Question\_id” is designed to store the blog question to which the answer belongs. “Father\_answer\_id” is designed to store the blog answers which the answer belongs to. Only one of the values between “Question\_id” and “Father\_answer\_id” is not null since the answer can only reply to the question or the answer. “Content” is designed to store the content of the answer. “like” is designed to store the amounts of users who like the answer. “Author\_id” is designed to store the author of the answer. “Create\_time” is designed to store the time period of the creation of this answer.

Attached\_file Some blogs or answers may have some attached files and those files will be stored in this entity. The entity has URL, corresponding\_question, corresponding\_answer, create\_time. “URL” is designed to store the URL of the file. “Corresponding\_question” is designed to store the blog question to which the file belongs. “Corresponding\_answer” is designed to store the blog answer to which the file belongs. Only one of the values between “Corresponding\_question” and “Corresponding\_answer” is not null since the file can only belong to the question or the answer. “Create\_time” is designed to store the time period of the creation of this file.

Attached\_picture Some blogs or answers may have some attached pictures and those pictures will be stored in this entity. The entity has URL, question, answer, group\_name, create\_time. “URL” is designed to store the URL of the picture. “Question” is designed to store the blog question to which the picture belongs. “Answer” is designed to store the blog answer to which the picture belongs. “Group\_name” is designed to store the group to which the picture belongs. Only one of the values among “Question”, “Answer” and “Group\_name” is not null since the picture can only belong to the question, the answer or the group.

User\_like\_answer In our online forum system, user can click the button to like the answer, so those information will be stored in this entity. The entity has id and answer\_id. “Id” is designed to store the id of the user. “Answer\_id” is designed to store the id of the answer. Therefore, each record means the user like the answer.

User\_like\_question In our online forum system, user can click the button to like the question, so those information will be stored in this entity. The entity has id and question\_id. “Id” is designed to store the id of the user. “Question\_id” is designed to store the id of the question. Therefore, each record means the user like the question.

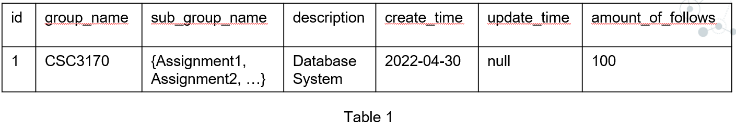
User\_follow\_question In our online forum system, user can click the button to follow the question, so those information will be stored in this entity. The entity has id and question\_id. “Id” is designed to store the id of the user. “Question\_id” is designed to store the id of the question. Therefore, each record means the user follow the question.

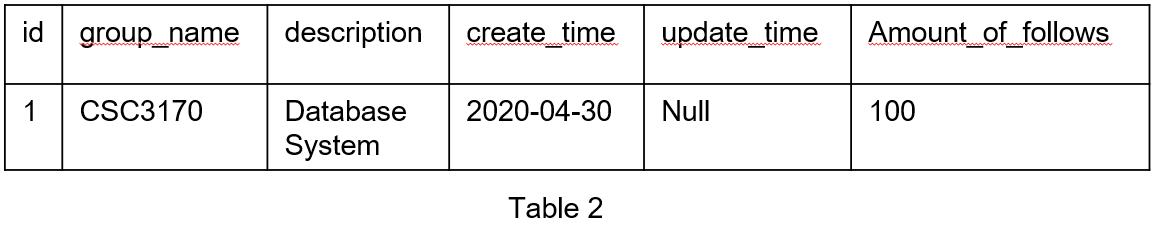
User\_follow\_group In our online forum system, user can click the button to follow the group, so those information will be stored in this entity. The entity has id and group\_name. “Id” is designed to store the id of the user. “Group\_name” is designed to store the name of the corresponding group. Therefore, each record means the user follow the group.

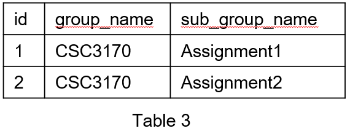
User\_view\_question In our online forum system, user can click the blog to view the detailed question, so those information will be stored in this entity. The entity has id, question\_id and time. “Id” is designed to store the id of the user. “Question\_id” is designed to store the id of the question. “Time” is designed to store how many times the user views the question.

*C. Schema & Normalization*

In our project, we spare a lot of effort on normalization. At first, we intend to reach the first normal form, so we reconstruct our database. For example, "Table 1" shows our initial design of the group table where has a tuple of sub group name. Therefore, we split the group table into group table (Table 2) and sub group table (Table 3). In this case, the first normal form is achieved.







Based on the first normal form, we also try to reach the second and third normal form. As a result, we create a unique id for each table to enable all nonprime attributes are fully functionally dependent on the primary key (id). Apparently, there does not exist nonprime attributes in our tables transitively dependent on the primary key (id). Therefore, the second normal form and third normal form are also implemented. As a result, the relational schema is shown below:

|  |
| --- |
| user(id, email, username, password, photo, major, grade, create\_time, update\_time) |
| group(id, group\_name, description, create\_time, update\_time, amount\_of\_follows) |
| sub\_group(id, group\_name, sub\_group\_name) |
| blog\_questions(id, title, author\_id, group\_type, sub\_group\_type, content, code, lang, content\_format, like, follow, hot, create\_time, update\_time, views) |
| blog\_answers(id, question\_id, father\_answer\_id, content, code, lang, content\_format, like, author\_id, create\_time) |
| attached\_file(id, url, corresponding\_question, corresponding\_answer, create\_time) |
| attached\_picture(id, url, question, answer, group\_name, create\_time) |
| user\_like\_answer(id, answer\_id) |
| user\_like\_quesition(id, question\_id) |
| user\_follow\_question(id, question\_id) |
| user\_follow\_group(id, group\_name) |
| user\_view\_question(id, question\_id, time) |

Some interpretation about foreign keys referencing:

* “group\_name” in sub\_group refers to “group\_name” in group: each sub group belongs to a group.
* “author\_id” in blog\_questions refers to “id” in user: each blog belongs to a user.
* “group\_type” in blog\_questions refers to “group\_name” in group: each blog belongs to a group.
* “sub\_group\_type” in blog\_questions refers to “sub\_group\_name” in sub\_group: each blog belongs to a sub group.
* “question\_id” in blog\_answers refers to “id” in blog\_questions: each answer belongs to a blog.
* “author\_id” in blog\_answers refers to “id” in user: each answer belongs to a user.
* “corresponding\_question” in attached\_file refers to “id” in blog\_questions and “corresponding\_answer” in attached\_file refers to “id” in blog\_answers: each file belongs to a question or an answer.
* “question” in attached\_picture refers to “id” in blog\_questions, “answer” in attached\_picture refers to “id” in blog\_answers and “group\_name” in attached\_picture refers to “group\_name” in group: each picture belongs to a question, an answer or a group.

*D. Index & Hashing*

1. SQL Function

This section is mainly about SQL functions. Many SQL requirements and their back-end SQL sentences and corre- sponding results will be demonstrated. They include opera- tional queries as well as analytic queries. All the functions are implemented in our system and can be accessed in the GUI.