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*

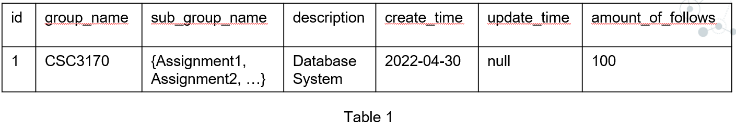
The main component of our project is the online forum database. There are three main entities in the database, namely “user”, “blog\_questions” and “blog\_answers”. The “user” entity stores the account information about the user, including their name, email address, password and etc., which is used for 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.

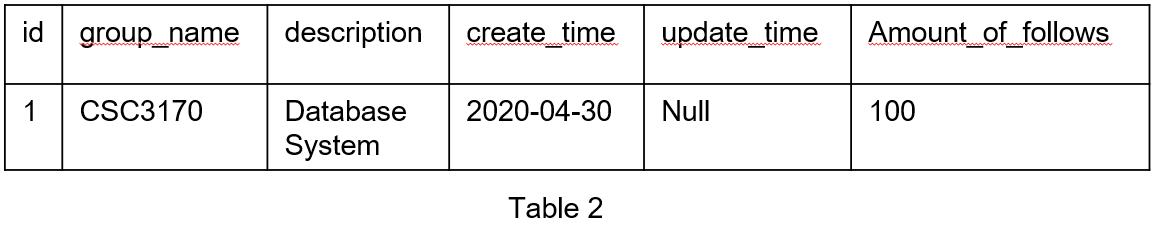
*B. Entity-Relation Diagram*

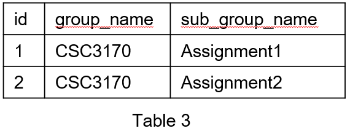
The ER-diagram of our system contains 12 entities. The entities are user, group, sub\_group, blog\_questions, blog\_answers, attached\_file, attached\_picture, user\_like\_answer, user\_like\_question, user\_follow\_question, user\_follow\_group and user\_view\_question.

*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.