CS-6083 Project 1 Report

*Jintian Zhu Duan Wang*

*jz2690@nyu.edu dw2201@nyu.edu*

**I. Introduction**

This report first briefly introduces our understanding of the project, then lists some of the most crucial requirements that this project must realize. The introduction part is followed by entity-relationship illustration and schema design, and using the schema design we created a database and inserted some fabricated data into it. The last part of this report includes some SQL querying examples, and the definition of some triggers. The first report of the project mainly covers the design of database using MySQL. The actual designing of the websitehasn’t yet been commenced, and this part will be introduced in the next deliverable.

The main task of the project is to design a crowdfunding websitethat allows user to start a fundraising campaign, or, as a sponsor, pledge money to some project. Similar to other database-backed websites, the website should allow basic manipulation on account, which includes login, logout and register, because in the project only limited information is needed for a user, account setting function is not going to be provided. As a crowdfunding website, this system should also provide interface for user to generate a new fundraising campaign and pledge certain project. When the minimum needed amount of money is raised for a project, all the sponsors are charged. Such projects are regarded as successfully funded. And after a project succeeds, the owner can inform the sponsors of their progress by publishing some description text or videos and photos, project owner can define the version number of current progress.In this project, sponsors will not get anything back as reward for pledging a project, but sponsors can comment on current progress. After a project has been completed, its sponsors can rate it. To provide some social networking features, user can follow other users, and like some projects. In this project we do not need divide user to different types as each user can start fundraising campaigns and pledge other projects. The use-case diagram of this project is shown below.



Figure 1. Use-case diagram

Based on the use-case, the database should contain an entity to store user accounts, an entity to store projects, an entity to store progresses of each project, and also an entity to store comment from each sponsor of a project progress. Further, user entity need have relations to other users to include the following information, and user can own a project, pledge a project, and like a project, so user entity should have different relations to the project entity. Further, a progress is attached to a project, and a comment is related to a user and a progress.

The next part will discuss our entity-relationship design in detail.

**II. E-R Diagram Design**

Entities shall include User, Project, Progress and Comment.

-User should have a unique user name (user name for login, could be an email address.) as primary key. Also, a login password is required. When role as a sponsor, the credit information should be required, and in other case, it is optional. When a user pledge, we will check his/her credit information is valid or not.

The relationship between two users is Follower, which is a multi-multi relationship.

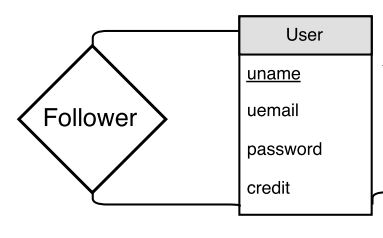


Figure 2. User and Follower

-Project should own a unique project id which is automatically produced by system. A project has one and only one owner: if a project is proposed not by a person but a group, for example, a band,then the band members can use one of their personal accounts or register a new account for their band to be the owner of the project. The project should include essential information for the funding: minimum amount to start the project; maximum raising amount; start date and end date of the fundraising campaign, and current amount sponsors have pledged to this project as an efficiency consideration. Also, information about the project itself includes: description (including txt, videos, images or url, this part will be further discussed and elaborated), project name, project tag (keywords for searching) and project status.

The relation between project entity and user entity can be owner and sponsor, or liker and being liked. As we discussed abovethe owner relationship is one-to-many relationship: a project only has one owner. And for the like relationship, a user can like many projects, and a project can be liked by many user, so it is a many-to-many relationship. Similarly, as for sponsor, the relationship is typically many-to-many relationship.

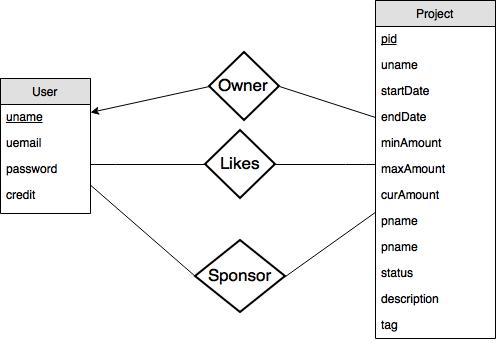


Figure 3. Project and its relationships

The sponsor relationship should containthe following additional information: the amount this sponsor pledges; the rate the sponsor marks (when project finishes), with default value null;a discuss attribute as an optional field.

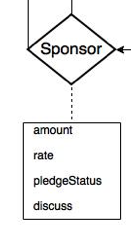


Figure 4. Sponsor

-Progress entity stores updatesof projects that are successfully funded. Progresses are ordered by version number. The version number is automatically formed by system. Each progress should contain some description, which can be a multi-media form, submittedby the project owner.

The relationship between progress and project is one-to-many relationship.

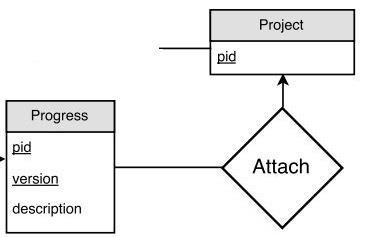


Figure 5. Progress and its relationship

-Comment entity is a weak entity which associates with sponsor and progress. And comment itself has a date information and comment content.

Each comment will associate one sponsor with a version of the project progress.

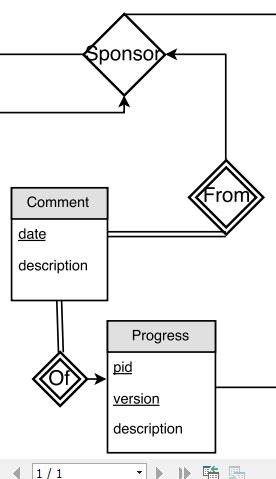


Figure 6. Comment and its relationships

Combining the schema design above, the final ER-Diagram is shown in Figure 7:

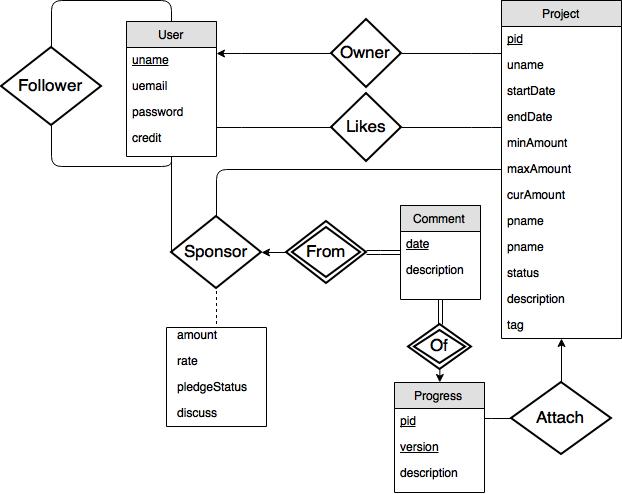


Figure 7. E-R diagram

**III. Schema Design**

Tables:

Four entities andtwo many-to-many relationships form the 7-tables schema:User, Follower, Project, Like, Sponsor, Progress, Comment.

*-User: uname, uemail, password, credit*

*-Follower:uname, funame*

*-Project: pid, unmae, startDate, endDate, minAmount, maxAmount, curAmount, pname, status, description, tag*

*-Like: pid, uname*

*-Sponsor: uname, pid, amount, rate, pledgeStatus, discuss*

*-Progress: pid, version, description*

*-Comment: uname, pid, version, date, description*

Since the Follower is a many-to-many relationship, two users’ primary keys should be contained. we name one of them as 'funame', meaning the follower's uname. Similarly, Like and Sponsor aremany-to-many relationships, so the primary key of user and project should also be contained. The primary key of Sponsor thus isuname and pid.The relationship of Progress and Project is many-to-one, so 'pid' should be included in primary key.Comment's primary key should contain Sponsor's primary key as well as Progress's primary key.

Other constraints:

In Project table, the curAmount cannot be larger thanmaxAmount. Status shall be relevant to curAmount, startDate and endDate, for instance, the status attribute should not be ‘FUNDING’ if current time is later that endDate. Because we can set the inserting query when a fundraising campaign is created, normally unreasonable tuple would not exist.

In Sponsor, the amount attribute cannot be larger than ‘maxAmount - curAmount’ of the corresponding project.

Analysis:

User contains the following functional dependencies: uname→uemail, uname→password, uname→credit. These three functional dependencies form the canonical cover of User table. Because for each dependency the left set (uname) is the super key of User, this schema is in BCNF.

There is no functional dependency in the Follower schema.

Of the Project schema, thesuper key also contains only one attribute, so for each of its functional dependency, the left set is the super key, thus this schema is in BCNF.

There is no functional dependency in the Like schema.

Sponsor contains the following functional dependencies: uname,pid→amount, uname, pid→rate, uname, pid→pledgeStatus, uname, pid→discuss. For each dependency the left set is the super key of the schema, thus Sponsor is in BCNF.

The canonical cover of Progress is only pid, version→description, and pid, version constitute the super key, so it is also in BCNF.

In the Comment table the only functional dependency of canonical cover is uname, pid, version, date→description, this schema is in BCNF as well.

**IV. Test with sample data**

Sample Data:

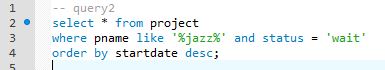
https://github.com/Dwan9/DatabaseCrowdFunding.git

Testing queries:

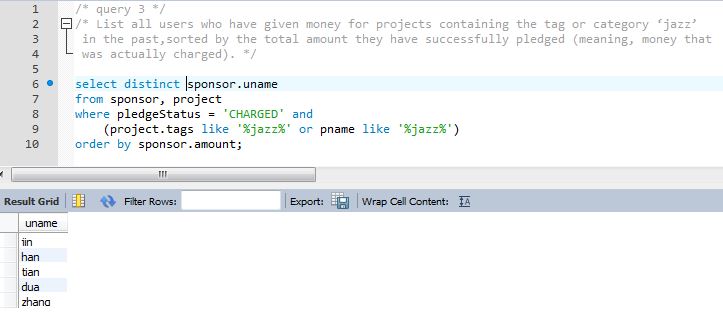
- Create a record for a new user account, with a name, a login name, and a password.

1.JPG

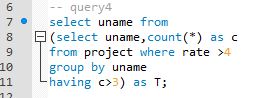
- List all projects that contain the keyword ‘‘jazz’’ and that are currently looking for funds, sorted indescending order by posting time.



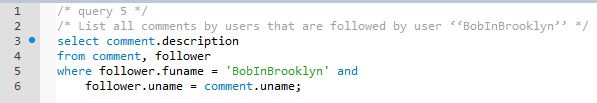
- List all users who have given money for projects containing the tag or category ‘‘jazz’’ in the past,  
sorted by the total amount they have successfully pledged (meaning, money that was actually charged).



- List all users who have completed at least 3 projects, and where each of their projects received anaverage rating of 4 stars or higher from its sponsors.



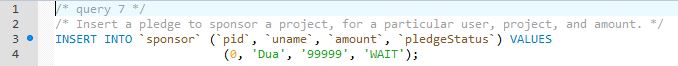
- List all comments by users that are followed by user ‘‘BobInBrooklyn’’.



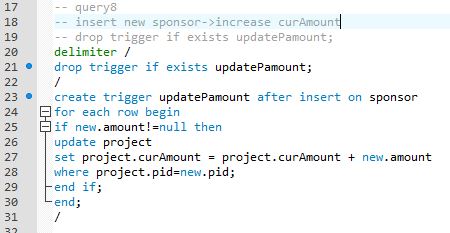
- Insert a new project for a particular user, with a name, description, and other needed info.

6.JPG

- Insert a pledge to sponsor a project, for a particular user, project, and amount.

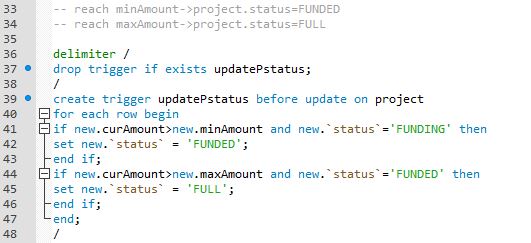


- Write queries for the end of a funding campaign. E.g., you could use triggers to detect when acampaign is fully funded or time is up; if successfully funded, generate charges to sponsors’ credit cards.

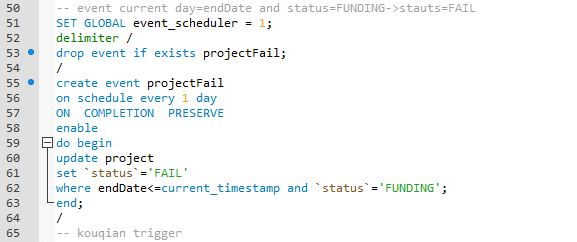


Other trigger defined for the database:

- Automatically alter a project’s status if current funding reaches minimum amount or maximum amount.



- Search every day for fundraising campaigns whose end date arrives, and set project status to ‘FAIL’.



- Every time a sponsor has rated a pledging experience, automatically recalculate average rating of the project and update it.



(continued in report 2)