# CSE 4/560 Project1: TinyPiazza

Due 23:59 03/09/2019 EST

March 1, 2019

## 1 Submission

*Failure to comply with the submission specifications will incur penalties for EACH violation.*

- What to submit: A zip file has to be submitted through the 'submit_cse460' (if you are CSE460 student) or 'submit_cse560' (if you are CSE560 student) submit script by 03/09/2019 11:59PM EST. Only zip extension will be accepted, please **don't** use any other compression methods such as tar or 7zip. You can submit multiple times, note that **only** the last submission will be kept on the server.
- Zip file naming: Use *ubit_proj1* (**NO SPACE!**) for the filename, for example: *jsmith_proj1.zip*, where *jsmith* is the ubit. The project is an **INDIVIDUAL** project, the filename should contain only one ubit.
- Sub-structure of zip file: On unzipping the zip file, there should be a folder named with your ubit *ubit_proj1*, under the folder *ubit_proj1*, there should be two files: (1) a .pdf report, and (2) a .sql SQL file.

## 2 Description

You are to design and implement the database schema for TinyPiazza, which is a course forum website provides simple functions, main functions of TinyPiazza are as the following:

- User management: user sign up, user login/logout.
- Course management: course basic information.
- Post management: managing posts in courses.
- User-Course relationship management: users can create/update or register courses.
- User-Post relationship management: users can create/update/delete posts.

Your DB schema must be able to support all the functions listed above. You are required to use E/R modeling (an online E/R diagram tool is `https://www.draw.io/`) to design and present your database, and map your E/R model to a relational database schema and implement the mapped schema using a RDBMS by a set of CREATE TABLE statements. A recommended RDBMS is:

- MySQL(`https://dev.mysql.com/downloads/`): you need to first install MySQL Community Server to use MySQL, then you'll need one of MySQL Shell and MySQL WorkBench for SQL development. If you need a GUI, you may prefer to use MySQL WorkBench.

You also need to briefly explain how your model supports all the required functions. The final product of this is a report (.pdf file), of which the structure will be given in section 4 in this description, and a SQL file (.sql file type) which contains all the CREATE TABLE statements you used for implementing your RDB schema.

**Note: you ONLY need to design and implement the database schema for the data needed to support the functions, you are NOT required to implement the functions.**

# 3 Requirements of functions

The specific requirements for each function of TinyPiazza are given in this section. **Note that you need to analyze and design the entity types, attributes and relationship types by yourself**, while this project description only gives some of them for clarifying the system requirements.

## 3.1 User management

- **User sign up**: **Users** sign up using their *email addresses*, users need to set their *passwords* and *display names*, where:
  - Email addresses uniquely represent the users.
  - Email address, password and display name are `strings` and can not be null.
  - One email address (i.e., one account) has only one display name, and one display name corresponds to only one email address.
- **Type of users**: TinyPiazza manages two types of users, namely **Student users** and **Teaching users**. A user can be both a student user and a teaching user, e.g., a TA.

### 3.2 Course management

As a course forum, TinyPiazza also store information of courses provided by different universities. The basic information of courses including:

- University name: e.g., University at Buffalo.
- Course number: e.g., CSE560.
- Course name: e.g., Data Models and Query Languages.
- Year: e.g., 2019.
- Semester: e.g., Spring semester.

University name, Year, Semester and Course number together form the key of courses. All of the attributes can not be null.

### 3.3 Post management

Users can post **posts** in a course. There are two types of **posts** in TinyPiazza, namely **question posts** and **answer posts**. The information of each type of posts are as the following:

- Question posts: users post question posts when they need to ask a question in a course. For question posts, we store the following basic information:
  - PostId: **within a course**, each question post has a unique postId, e.g., 1, 2, 3, etc.
  - Title: the title of the post, which is a string consists up to 100 charaters. Title can not be null.
  - Description: the question post's content, description is stored as a string.
  - Time: the posted time of a question post, can not be null.
- Answer posts: users can answer questions in question posts by posting answer posts for the question post. Note that as answers are for questions, each answer post is associated with a question post, i.e., an answer post contains an answer for a question post. We store the following basic information of an answer post:
  - Answer: the answer post's content, answer is stored as a string, can not be null.
  - Time: the posted time of an answer post, can not be null.

### 3.4 User-Course relationship management

**Teaching users** can *create* and *update* many courses, a course can be created by exactly one teaching user, but can be updated by many teaching users. **Student users** can *register for* many courses. A course can have many students.

### 3.5 User-Post relationship management

A user can post/update many posts. A post can be created by exactly one user, as well as can be updated by the post creator only.

## 4 Report template

Here's a template of the report, your report **must contain** each of the following sections, you can extend this template to include any necessary sections for your design.

- E/R schema: in this section you need to introduce your E/R schema and include the picture of your E/R schema.
- Relational database schema: in this section, you must
  - discuss briefly how you map the E/R schema to your relational database schema. If any design choice is made in the mapping process, illustrate and explain it briefly.
  - discuss briefly how your relational database schema satisfies all the requirements listed in section 3.
- Further discussion: in this section, you need to discuss the advantages and disadvantages of your design.

## 5 Extra credit (4 pts)

Consider the schema $R(ABCD)$, the set of functional dependencies $F = \{A \rightarrow BCD\}$ and the decomposition $(R_1, R_2) = (ABC, ABD)$.

Questions:

1. Are $R_1$ and $R_2$ in BCNF? Which dependencies are preserved by the decomposition?

2. If your answer to the previous question was "Both are in BCNF", how would you explain the following situation? An instance $r$ of $R$ contains a single tuple $(frank, 100K, mary, rochester)$. In the decomposition $(R_1, R_2)$, the instance $r_1$ of $R_1$ consists of the tuple $(frank, 100K, mary)$ and the instance $r_2$ of $R_2$ consists of the tuple $(frank, 100K, rochester)$. The connection between $frank$ and $100K$ is represented twice, creating a redundancy and a possible update anomaly. BCNF is supposed to prevent this.

Justify your answers. Use an extra section in your report to include your answers for this question.