INFO20003 Semester 2, 2020

Assignment 2: SQL
Due: 6:00pm Friday 25 September 2020
Submission: Via LMS https://canvas.lms.unimelb.edu.au/

Weighting: 10% of your total assessment

Unichat Startup Database

You and a group of fellow undergrads have created the UniChat startup company. The company's goal is to run a popular online discussion forum for students at the University of Melbourne.

The system has two kinds of users: students and lecturers. Most user attributes are straightforward personal information and are listed in the ER diagram below. "Karma" is a score awarded for being active in the system and posting popular posts.

Discussions are organized into forums, each of which is about a particular topic. Individual forums can be opened and closed by lecturers. When a forum is closed it is not deleted, but simply marked as closed and hidden from users. Users can post a text message into any open forum and can comment on existing posts. Comments are stored in the same table as posts and are connected to their parent posts via a unary relationship. Users can also click a "Like" button which is attached to every post or comment; this is recorded so it can be represented to other users. Because posts and comments are the same entity, questions that refer to "posts" mean "posts and/or comments" unless we specify "top level posts". Note that in the model, "top level posts" have a FK to the forum they are posted in, but comments have NULL for forum FK (since they're not a "forum" post).

Students (but not lecturers) can add each other to friend-lists, which are used to influence the ordering of posts and the other website behaviour. When one student sends a friend-request to another, the latter can reject or accept the friendship. If the latter accepts, the pair are now friends. Note that friendship is a symmetric relation: if A is a friend of B, then B is a friend of A, whether A sent the friend request to B or vice-versa. Once a pair of users are friends, either may later unfriend the other, in which case the friendship ends for both of them. Each time a friend-request is sent, rejected, accepted or unfriended; this is recorded in the FriendOf table by adding a timestamp in the appropriate column. We calculate who a given student's friends are by querying this table: a user's "friends" means their current friends, not those where the friendship has ended or not begun.

The Data Model

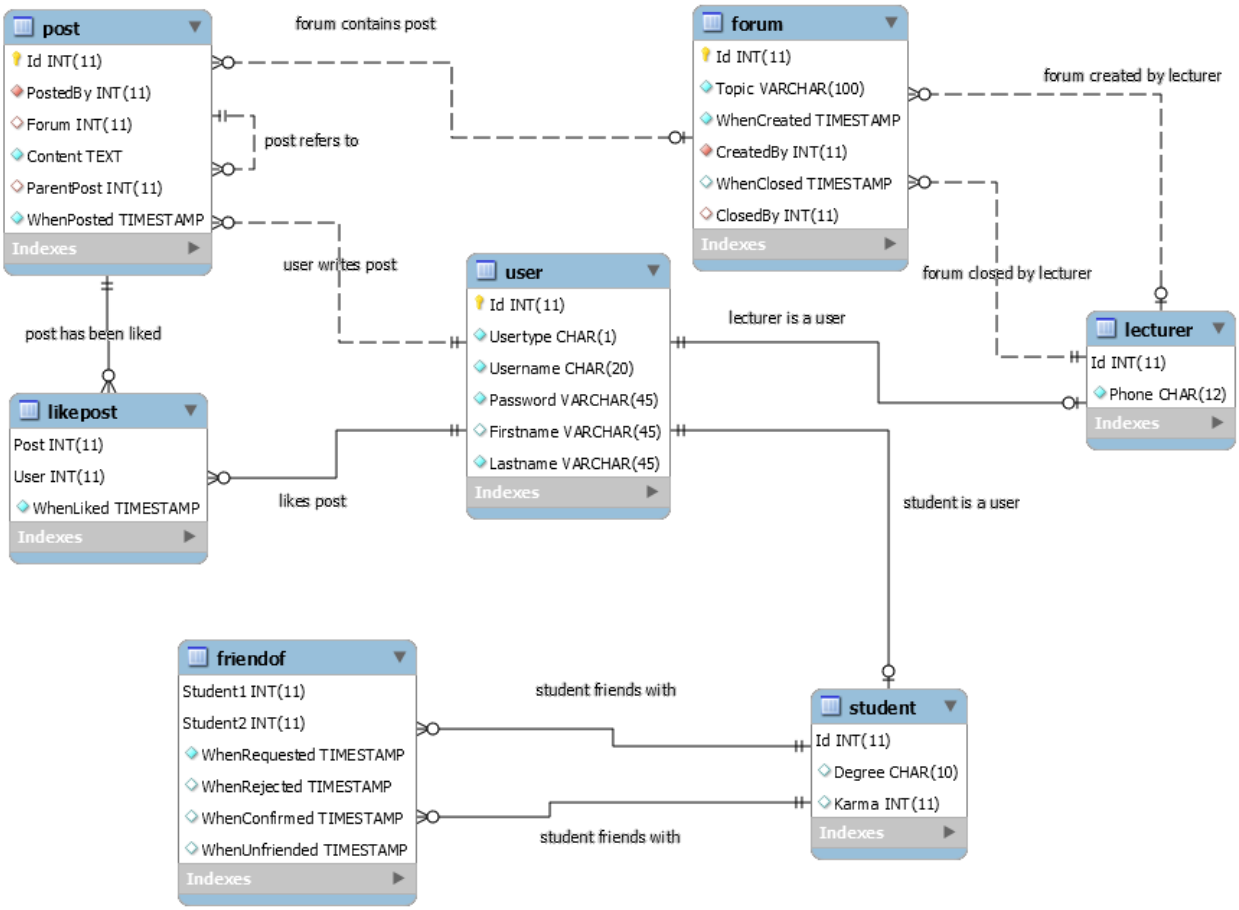


Figure 1: The ER Model for Unichat Database

Assignment 2 Setup

A dataset is provided against which you can test your solutions to the assignment. To set up the dataset, download the file **unichat_2020.sql** from the Assignment on Canvas and run it in Workbench. This script creates the database tables and populates them with data.

The script is designed to run against your account on the Engineering IT server (info20003db.eng.unimelb.edu.au). If you want to install the schema on your own MySQL Server installation, uncomment the three lines at the beginning of the script.

***Note: Do NOT disable full_group_by mode when completing this assignment**. This mode is the default, and is turned on in all default installs of MySQL workbench, you can check it is turned on using the command "SELECT @@sql_mode;", it should return a string containing "full_group_by". When testing, our test server WILL have this mode turned on, and if your query fails due to this, you will lose marks.*

The SQL Tasks

In this section are listed 10 questions for you to answer. Write one (single) SQL statement per question. Subqueries and nesting are allowed within a single SQL statement – however, you may be penalized for writing *overly* complicated SQL statements. DO NOT USE VIEWS (or 'WITH' statements/common table expressions) to answer questions. *Important: please try and ensure that your query returns the projected attributes in the same order as given in the question to help us with marking; the names of columns aren't important.*

1. List all the forums created and closed by the same lecturer. Your query should return results of the form (forum ID, topic, lecturer ID who created/closed them). **(1 mark)**

2. For each lecturer show how many forums they have created. Your query should return results of the form (Lecturer ID, their full name, the number of forums created by them). **(1 mark)**

3. List all the users who never posted a top-level post. Your query should return results of the form (userID, username). **(1 mark)**

4. Display the post ID of the top-level posts with no comments and no likes. Your query should return results of the form (postID). **(2 marks)**

5. List the post ID, content and count of posts with the most number of likes. Display all such posts in case of a tie. Your query should return results of the form (postId, content string, number of likes). **(2 marks)**

6. Which top-level post has the longest length? Display its length, its content, topic of the forum where its posted and the first and last name of the user (combined as full name) who posted it. Display all posts with the longest length in case of a tie. Your query should return results of the form (length, content string, topic of forum, user's full name). **(2 marks)**

7. Display two students who remained friends for the shortest duration. List IDs of both students and the number of days they remained friends for. Display all such pairs in case of a tie. If A and B are such friends, it is sufficient to display only A and B (no need to display the symmetrical solution B and A). Your query should return results of the form (student1ID, student2ID). **(2 marks)**

8. For each user liking a post, calculate how many other users have liked the same post. Display the user ID, the number of other users who are liking the same post and post ID. Your query should return results of the form (userID of person who liked post, count of other users liking the post, postID). **(3 marks)**

9. Let us call the student who has the greatest number of likes on the website (across all posts) the "most popular student". List userIDs of all the currently confirmed friends of the "most popular student", who are also doing the same degree as the "most popular student". Assume there are no ties (i.e., only one person is most popular). Your query should return results of the form (userID). **(3 marks)**

10. List all top-level posts by a student which did not receive a reply from the lecturer who created that forum within 48 hours, or are currently unanswered. Your query should return results of the form (post ID, the time when the post was created). **(3 marks)**

Submission Instructions

Your submission will be in the form of an SQL script. There is a template file on the LMS, into which you will paste your solutions and fill in your student details (more information below).

This .sql file should be submitted on Canvas by **6pm** on the due date of **Friday 25 Sept**. Name your submission as 987654.sql, where 987654 corresponds to YOUR student id.

Filling in the template file:

The template file on the LMS has spaces for you to fill in your student details and your answers to the questions. There is also an example prefilled script also available on the LMS. Below are screenshots from those two documents explaining the steps you need to take to submit your solutions:

<table>
<tr><th>Step</th><th>Example</th></tr>
<tr><td rowspan="2">1. At the top of the template, you'll need to replace "XXXXXXXX" with your student number and name</td><td>Template
<pre>-- Your Name: XXXXXXXX
-- Your Student Number: XXXXXXX</pre></td></tr>
<tr><td>Example Filled in
<pre>-- Your Name: Colton Carner
-- Your Student Number: 693281</pre></td></tr>
<tr><td rowspan="2">2. For each question 1-10, place your SQL solution in between the "BEGIN QX" and "END QX" markers. Ensure each query is terminated with a semicolon ";"</td><td>Template
<pre>--_____
-- BEGIN Q1

-- END Q1
--_____</pre></td></tr>
<tr><td>Example Filled in
<pre>--_____
-- BEGIN Q1

-- **This is an example of a comment which will not be executed
-- **(comments are not NEEDED, but if your query is complex you can leave some)

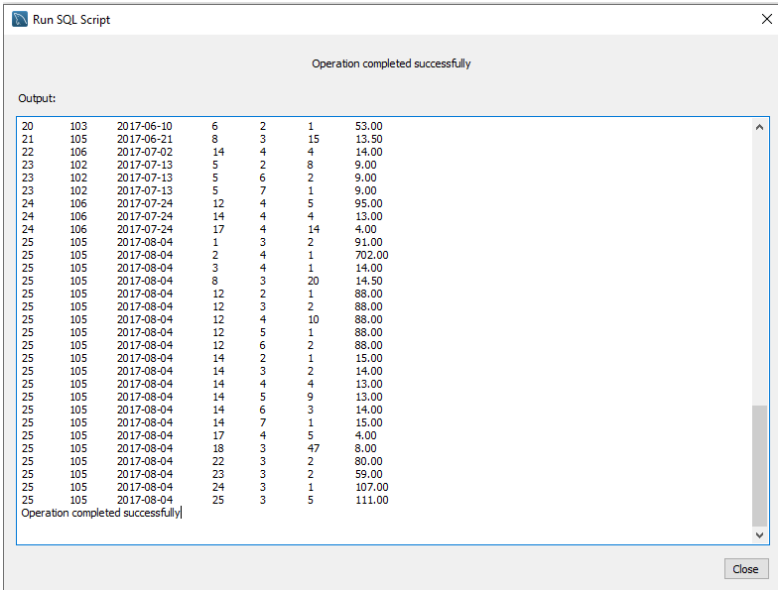
-- **Below is an example of how you would enter your answer:
SELECT *
FROM delivery
NATURAL JOIN deliveryitem
WHERE supplierid = 101;

-- **It's OK to add more space in between the 'BEGIN QX' and 'END QX' markers
-- **for each question, just don't DELETE the markers!

-- **Make sure you fill out your name + student num at the top of the document

-- **After reading / understanding these comments in the Q1 section
-- **(comments starting with '**'), feel free to delete them
-- **(don't delete lines without **)

-- END Q1
--_____</pre></td></tr>
</table>

3. Test that your script is valid SQL, by running it from MySQL Workbench (file > “Run SQL Script”). Make sure to select the default schema as the Unichat schema you imported when prompted	Running the example filled in template provided, using schema we’ve used in labs (dep-store) results in ‘success’. (You’d run yours on the schema for the A2 assignment) 

Requesting a Submission Deadline Extension

If you need an extension due to a valid (medical) reason, you will need to provide evidence to support your request by *9pm, Thursday 24 Sept*. Medical certificates need to be at least two days in length.

To request an extension:

- Email Farah Zaib Khan (farah.khan@unimelb.edu.au) from your university email address, supplying your student ID, the extension request and supporting evidence.
- If your submission deadline extension is granted you will receive an email reply granting the new submission date. Do not lose this email!

Reminder: INFO20003 Hurdle Requirements

To pass INFO20003, you must pass two hurdles:

- **Hurdle 1:** Obtain at least 50% (15/30) or higher for the three assignments (each worth 10%)
- **Hurdle 2:** Obtain at least 50% (35/70) or higher for the combination of quizzes and end of semester exam

Therefore, it is our recommendation to students that you attempt every assignment and every question in the exam.

GOOD LUCK!