CS 3200: Database Design / Prof. Rachlin
Homework 3: Twitter -- Database Design and Validation

PROBLEM DESCRIPTION:

Let's explore a very practical and real-world database design problem: implementing a Twitter-like service. Twitter is a fascinating case-study in database design and big data architectures. Originally, twitter was backed by a MySQL database, but scalability issues led them to implement a non-relational NoSQL approach.

In this assignment you will start with a high-level description of the Twitter service and then:

a) Create a conceptual / logical model using the MySQL Workbench modeling tool

b) Write a script that constructs your database schema with hand-crafted CREATE TABLE statements and that INSERTs some fake data into each of the tables of your schema. (*Important: Do NOT use MySQL's Forward Engineering feature to create the tables – the resulting CREATE TABLE statements are unnecessarily verbose and sometimes incorrect!)*

c) Write some queries to validate your database design.

PART A. Conceptional Modeling (40 Points)

Use the MySQL workbench to create an EER-Diagram for the following domain description. Please not that this may not be exactly like the real Twitter. Don't make your model any more complicated than it needs to be. Save your model as an .MWB file.

Twitter is a social network where users post tweets. A tweet is a string of text at most 160 characters long. Each tweet has a timestamp and is created by one and only one unique user. Users follow other users. Tweets may contain hashtags (words proceeded by a '#' symbol) and users need to be able to efficiently search for tweets that contain specific hashtags. Users can indicate that they "like" a particular tweet they find, even if the tweet isn't by one of the users that they normally follow. A user can be a person or an organization. Users have a full name "John Doe" or "Northeastern University", a twitter handle (@JohnDoe) an email address, password, and an optional short profile (255 characters max). When users open up the twitter app, the see the most recent tweets by the users they follow. This list of tweets is called the user's "home timeline." (The home timeline is something computed on the fly by Twitter – you do NOT need a table for the home timeline.) Users can mark their profile as hidden so that it doesn't show up in any user recommendations and all of their tweets will instantly disappear from the home timelines of all followers.

PART B. Logical Modeling (30 Points)

Construct a set of tables with reasonable column names and datatypes that can manage the content and data operations of the above conceptual model. For example, you'll want to get started by having a USER table with a *user_id* and a TWEET table with a foreign key (*user_id)* referencing the USER table to identify who wrote the tweet, and so on. Add appropriate primary key and foreign key constraints. Write a series of INSERT statements that populate each table with perhaps 5-10 rows of data. The data can be entirely made up, but each record should be unique and the data should be internally consistent.

PART C. Database Validation (30 Points)

Write a SQL query that answers the following question. (Since we haven't learned about JOINs, all your queries will be against a single table.)

a) Which user has the most followers? Output just the user_id of that user, and the number of followers.

b) For one user, list the five most recent tweets by that user, from newest to oldest. Include only tweets containing the hashtag "#NEU"

c) What are the most popular hashtags? Sort from most popular to least popular. Output the hashtag_id and the number of times that hashtag was used in a tweet. (Depending on how you design your database, you do not need to output the hashtag itself as this may be stored in another table and we haven't yet learned out to do joins!) Rank your output by number of occurrences in descending order.

d) How many tweets have exactly 1 hashtag?

e) What is the most liked tweet? Output the tweet attributes.

SUBMIT:

a) Your conceptual model (.MWB) for Part A.
b) A script (.SQL) that builds your database, populates it with sample data, (Part B) and answers each question from Part C.