Revision: 10/07/2020
Please watch for updates on the project description

COMS 363 Fall 2020

Project: What did our state legislators tweet during the 2016 and 2020 presidential election years?

Submission due date: Friday November 20, 2020 by 11:50pm
Percentage in your final grade: 25%
Total points: 200
Number of students per team: Up to **two students** per team
Important: The amount of work is same regardless of the number of team members as teamwork also incurs communication overhead.
Per course policy, late submissions are not graded.

Learning objectives:

These objectives have an ultimate goal to increase students' confidence in their skills to manage a medium size real-world dataset.

1. Practice relational database design that reduces redundancies.
2. Practice ETL (Extract-Transfer-Load) to bring data into a relational DBMS. This process is often very time-consuming and involves eliminating inconsistency in the data.
3. Practice physical database design and query rewriting to improve query performance.
4. Practice writing Java database applications.

Overview

This project asks for development of a relational database and a Java database application for a client who is interested in learning about Twitter communication made by state legislators. The database is to store tweets issued by state legislators and presidential candidates around the 2016 and 2020 presidential elections. The Java database application accesses data from MySQL backend using MySQL Java Database Connectivity (JDBC) library. The tasks are as follows.

- Design a good relational database from a given ER diagram. The design needs to minimize unnecessary redundancy and to ensure that all constraints are enforced including total participation without a key constraint.
- Import the data from the csv files into MySQL Server and clean up inconsistency in the data.
- Write SQL queries for the questions specified in Table 1.
- Fine-tune these SQL queries to run faster and report the performance difference before and after the optimization.
- Implement Java application that sends SQL statements to the backend DBMS, receives the results, and display them on screen.

What is/will be provided for you:

1. ER diagram for the below database requirement
2. Csv files of tweets and related data such as hashtags, urls, and user accounts.
3. Documentation, videos, and SQL script examples to help you complete various tasks about the project.

4. Query output examples for the queries in Table 1 to be provided by October 20.
5. Lecture notes, pre-recorded videos, and Java code examples about JDBC.
6. Lecture notes, pre-recorded videos, and Java code examples using JDBC API to tell the transaction manager of the DBMS to prevent interference from concurrently running programs to ensure that the correct results of the programs are persistently stored in the database.

Database Requirement

- A tweet has the following properties: id, retweet_count (the number of retweets of this tweet), tweet text, created_at (number of milliseconds since 1/1/1970) in which the tweet was posted. The id is unique among all tweets. A tweet must have the user who posted it. If the posting user is deleted, all tweets posted by the user must also be deleted. A retweet has the same tweet text, but has a different id from the original tweet.
- A tweet may have zero or more hashtag in it. Each hashtag has a unique name and must be used in at least one tweet. A tweet has zero or more URLs. Each URL has a unique URL address and must appear in at least one tweet.
- A tweet may mention zero or more user accounts. A user account can be mentioned in zero or more tweets.
- A user account has the following properties: name, screen name, the number of followers, the number of people this user follows, sub_category, category, and the state the user lives. The name property can be null, but the screen name cannot be null and must be unique among all users. The sub_category indicates the party to which the user belongs. The values of this attribute are “GOP”, “Democrat”, “na”, or null. The values of the category attribute are “senate_group”, “presidential_candidate”, “reporter”, “Senator”, “General”, or null. A user lives in at most one state. Presidential candidate accounts are not associated with any state. We use “na” as the value of the state attribute for the user account without an associated state.

Check the data from the given csv files to understand the kind of data to be maintained and for choosing the data type appropriate for each attribute. Furthermore, there are some inconsistencies in how the names of the states are stored. For instance, Florida appears as “Florida” in some rows and as “FL” in some other rows. The inconsistency is the result of using No-SQL DBMS that does not require creation of the schema beforehand.

The ER diagram from the above database requirement is given below.

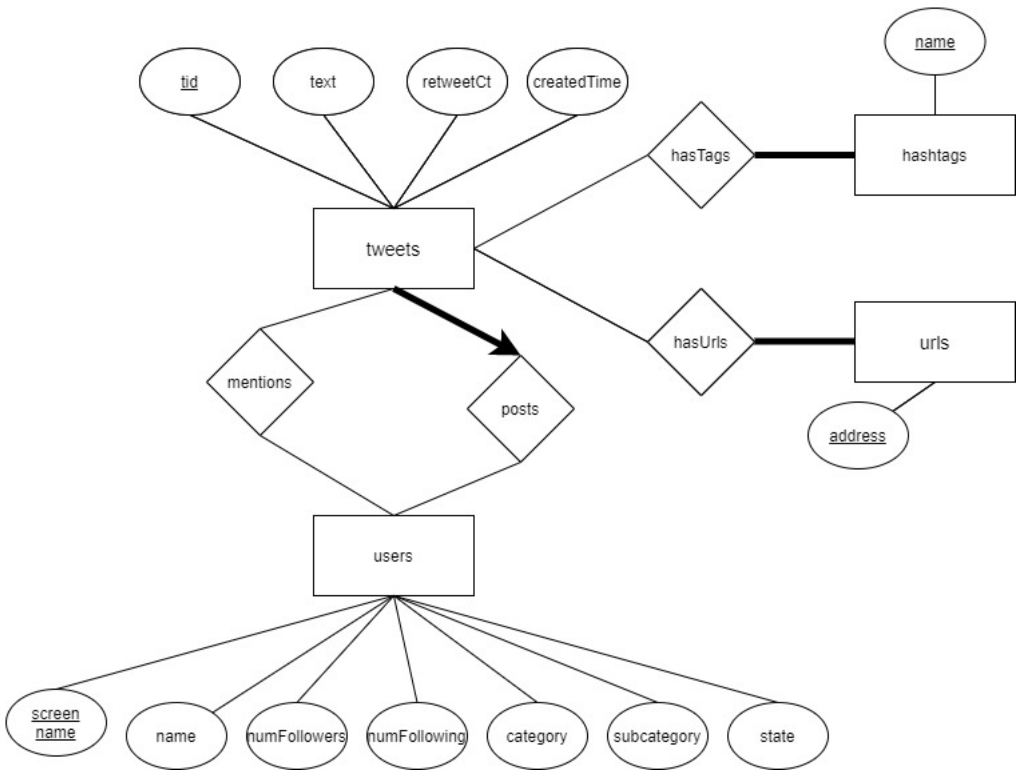

Figure 1: ER diagram of the database requirement

Table 1: Required functionalities of the Java database application. There are six queries and two insertion and deletion statements.

ID	**Description**
Q3	Find *k* hashtags that appeared in the most number of states in a given year; list the total number of states the hashtag appeared, the list of the distinct states it appeared (FL is the same as Florida*), and the hashtag itself in descending order of the number of states the hashtag appeared. **Input:** Value of *k* and year (e.g., 2016) **Rationale:** This query finds *k* hashtags that are used across the most number of states, which could indicate a certain agenda (e.g., education, healthcare) that is widely discussed. **Hint:** Use group_concat() function to create a list *This requires making sure that each state is represented using one name, either FL or Florida
Q7	Find *k* users who used a given hashtag in a given state in a given month of a given year. Show the count of tweets posted with that hashtag along with the user's screen name and category in descending order of the tweet count. **Input:** hashtag, state name, value of k, month, year **Rationale:** This is to find the users who used a given hashtag most often in a given state. These users could influence a certain policy agenda within the given state.
Q9	Find top *k* most followed users in a given party. Show the user's screen name, the user's party, and the number of followers in descending order of the number of followers. **Input**: Value of category (e.g., 'GOP', 'Democrat') **Rationale**: This query finds the most influential users measured by the number of followers
Q16	Show names and categories of k users along with the tweet text, retweet count, and the url used by the user a given month of a given year in descending order of the retweet count **Input**: value of k **Rationale**: This query finds the most influential tweets along with the user who posted and the urls used by the user.

Q18	Find *k* users who were mentioned the most in tweets of users of a given party in a given month of a given year. Show the user's screen name, user's state, and the list of the screen name of the user(s) who mentioned this user in descending order of the number of tweets mentioning this user. **Input:** Values of k, sub-category (e.g., 'GOP'), month, year
Q23	Find *k* most used hashtags with the count of tweets it appeared posted by a given sub-category of users in a list of months. Show the hashtag name and the count in descending order of the count. **Input:** Values of k, sub-category (e.g., 'GOP'), a list of months (e.g., [1, 2, 3]), year=2016
I	Insert information of a new user into the database. **Input:** All relevant attribute values of a user
D	Delete a given user and all the tweets the user has tweeted, relevant hashtags, and users mentioned **Input:** screen name of the user to be deleted Must check that a user is valid before doing so. If the user's screen name is not valid, abort the transaction.

- The value of *k* is between 1 and 100.

Deliverables

1. Deliverables on Database design (40 points)

Submission per group

- (10 points) ProjectDDL.sql that consists of SQL DDL statements that create the relational schemas and integrity constraints. Choose data types for your attributes that ensure that all the data are inserted correctly. Do not include any trigger code in this script since triggers slow down the loading of the data. This file must include SQL statements that drop the database of the same name if it exists and creates the database if it does not exist, respectively. ProjectDDL.sql must not include any physical database design choices such as indexing and or which MySQL engine to use.

  Naming convention: Name your database "group<id>" where <id> is replaced by the id of your project group according to Canvas. Name the relations according to the names of the corresponding entity set or the relationship set. Name the attributes of the relations according to the names of the corresponding attributes in the corresponding entity set or relationship set in the ER diagram in Figure 1.

- (14 points) ProjectInsert.sql has code that inserts the data from the given csv files into MySQL Server. Use the bulk loading method (https://dev.mysql.com/doc/refman/8.0/en/load-data.html) using LOAD DATA INFILE statement. Import only the values of the attributes that you need from the csv files into MySQL Server 8.0. See the example "example-load-infile.sql".

  The bulk loading method is a fast way to load the data. However, it is known to cause some strange issues for some systems. If you have trouble to use the method, use the import wizard like we did to import data into emp_work_dept. The import wizard takes much longer time, especially on a cloud virtual machine. Once you get the data loaded into the database, create the SQL dump file and rename it ProjectInsert.sql. Watch the video on how to generate a SQL dump file of a table that has only the insert statements of all the tables into one script file.

- (12 points) EnsureDataConsistency.sql has SQL code that cleans up data inconsistency. In other words, the code ensures that each state has one name to represent it. For instance, ensure that all the users who live in Florida has the attribute value either "FL" or "Florida". This script also includes the code that creates any trigger required to enforce the total participation without the key constraint.

- (4 points) CreateDBUsers.sql that has an SQL statement that creates a user "cs363@%" with the standard authentication method. The @% after cs363 allows this user to access from any remote machines. This user has the privilege only to view, drop, create, insert, and delete the data in your MySQL database. The best practice for writing a secured database applications is to use a non-root database account and limit the privilege of that account to the relations that they need access. Do not use root account in your code.

IMPORTANT

- All scripts must run successfully. Graders will run ProjectDDL.sql, ProjectInsert.sql, EnsureDataConsistency.sql, and CreateDBUsers.sql in this order.
- The design at this stage must aim to reduce unnecessary redundancy as the main objective.

2. Deliverables on query optimization (40 points)

Submission per group

- Before.sql that has SQL queries before query optimization. Give the SQL statements with specific values for the experiments that result in the performance reported in the column "Before optimization" in Table 2.

  You won't be able to see query execution plans if the queries are inside a stored procedure.

- PhysicalDesign.sql that has SQL statements that implement physical database design choices (e.g., creating indexes) to reduce query response times for which queries. If you do not use any index, put a comment in this file that indicates so. If you do use a method that increases the size of the database buffer pool, put a comment in this file to indicate how much the database buffer size you use for the improvement.

  IMPORTANT:
  You can apply one optimization method for at most two queries. For instance, if you increase the database buffer pool size as an optimization method for Q3 and Q7, you will have to use a different optimization method for the rest of the queries.

- After.sql that implements optimized SQL queries (possibly using the indexes created in PhysicalDesign.sql). In this file, give SQL statements and the specific values you use for the experiments that result in the performance reported in the column "After optimization" in Table 2.

- PerfSQL.pdf that has Table 2 filled up

Table 2: Average server execution time reported by MySQL server over 5 runs for each query

Buffer pool size: ___________ GBytes

Query ID	Optimization method	Before optimization (ms)	After optimization (ms)
Q3			
Q7			
Q9			
Q16			
Q18			
Q23			

3. Deliverables on Java database application (80 points)

Ideally, you want to use the optimized queries in After.sql. However, for ease of development, you can also use stored procedures or queries in Before.sql.

One submission per group

Source code: Project folder with Java source files. Make sure that your application is running successfully before submitting it. Comment the code and provide the names of the authors for each source file.

README file: This file documents what you need the teaching staff to know in order to install your code and run your code successfully. It should include the password for the cs363 account.

Grading criteria on the application

- The application must take advantage of MySQL as much as possible. The application that brings irrelevant data from the database and uses Java code to filter out irrelevant data will lose major points even the output is correct.
- The application must ensure that each functionality in Table 1 is in its own transaction.
- The application must use parameterized SQL queries.
- Each Java file must include the author(s) of the file/functions and must include some comments on the code.
- The application must use cs363@%1 account to make database connections. The password for the account must not be in the top 25 most popular passwords according to this site.
    1. https://www.welivesecurity.com/2018/12/17/most-popular-passwords-2018-revealed/

User interface friendliness is not graded, but it needs to be obvious for application users to perform the required functionality. Console applications are also accepted. Provide a README file how to run your code.

4. Deliverables on individual contributions (20 points)

Each team member must be involved in every part of the project.

- (10 points) Team work log in text format that provides the meeting time, meeting duration, present team members, tasks assigned; the team will also lose points if starting the project in the last two weeks of the semester.

Example TeamWorklog.txt

```
Week of Feb. 11, 2020

•   Team received the project description and the data
•   Team meeting (1 hour) and individual tasks assigned
    •   James's task:
            o   Design the ER diagram;
            o   Check the schemas designed by Liam
    •   Liam's task:
            o   Proofread the ER diagram
            o   Design the relational schemas from the ER diagram
```

- (10 points) Work logs of individual team members in text format; members who do not submit their individual work log will lose points on their contribution. Name the file as <netid>Worklog.txt. Each log must show continuation of the member effort throughout the project.

Example of individual Worklog.txt

```
James:

Week of Feb. 11, 2020

•   Designed the ER diagram (4 hours). Sent it to Liam for review.
```