Revision 10/11/2022

COMS 363 Fall 2022

Homework 3

Percentage in your final grade: 12%
Maximum score for the assignment: 200 points
Objectives:

1. Write a database application in Java using Java Database Connectivity (JDBC) with transaction support. Relational DBMS provides transaction support via JDBC Connection class method calls: setAutoCommit, commit, and/or rollback, and setTransactionIsolation.
2. Practice calling parameterized SQL statements.
3. Practice writing stored procedures.

Submission requirements

Submit A zip file named <netid>HW3.zip with the following files: **findPopularHashTags.sql**, **mostFollowedUsers.sql**, **insertUser.sql**, **deleteUser.sql**, and **tweetsdb.java**. Replace <netid> with your university netid. Each file needs to include as a comment your full name.

Preparation

For security purposes, when writing a database application, it is a good practice to use a user who has only the necessary privileges to access the database. Users who do not need to access this database do not need to see its existence. Furthermore, it is a good practice to avoid hard coding the database username and password in the application code itself. The username and the password can be provided by the application user or stored in an encrypted file and are decrypted at the time when the application needs to connect to the database.

In this assignment, your Java application uses coms363 as the username to access a local database on the same computer this application runs. Furthermore, the application asks for the username and password before allowing access to the database. See an example in the JDBCTransactionTester.java code for Week 8 on Canvas.

1. Create the practice database using the given practice_db_dump.sql.

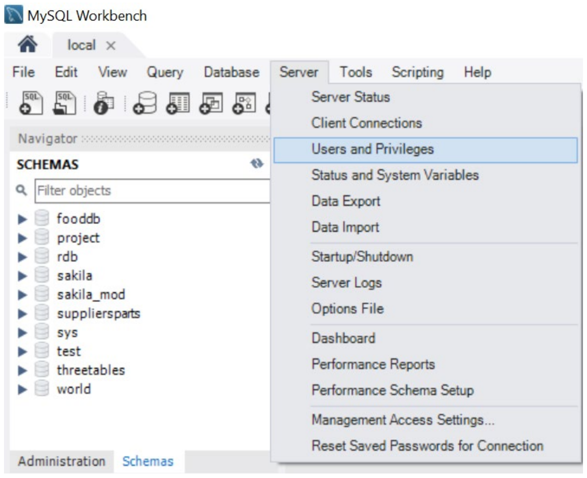
More explanation about the database: A Twitter user account has the following properties: name, screen name, the number of followers, the number of people this user follows, subcategory, category, and the state the user lives. The screen name cannot be null and is unique among all users. The subcategory indicates the political 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. Presidential candidate accounts are not associated with any state. The value "na" is used for the state attribute for the user account without any associated state.

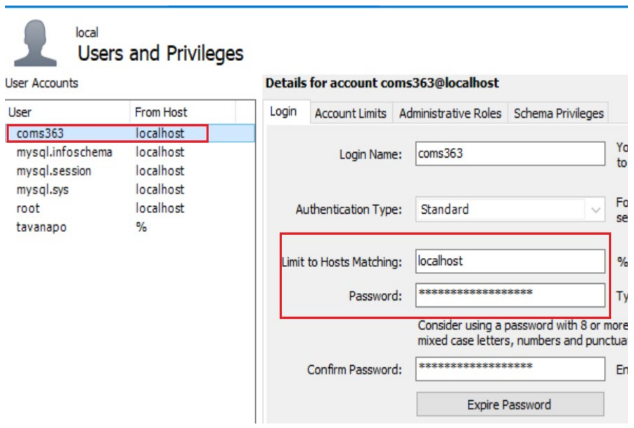
2. As root, execute the provided createuser.sql in your MySQL Workbench. The script does the following.

- Drop an existing user named 'coms363'@'localhost'.
- Create a new user named 'coms363'@'localhost' with the password "363F2022". The @'localhost' is not the username but a modifier that limits the access of the coms363 user from the computer where your database server is installed.
- Grant, coms363, all privileges (i.e., select, insert, update, delete rows, create tables, alter tables, references, and create indexes as well as create temporary tables and lock tables) to the "practice" database created by running practice_db_dump. This user must not see other databases in your local database server.

  If the script runs correctly, you will see this new user created when choosing the Server/Users and Privileges.

Revision 10/11/2022





User	From Host
coms363	localhost
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
root	localhost
tavanapo	%

Details for account coms363@localhost

Login Name: coms363

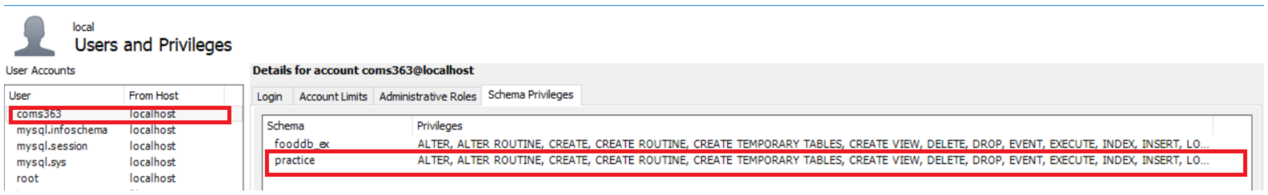
Authentication Type: Standard

Limit to Hosts Matching: localhost

Password: ******************

Consider using a password with 8 or more mixed case letters, numbers and punctua

Confirm Password: ******************



User	From Host
coms363	localhost
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
root	localhost

Details for account coms363@localhost

Schema	Privileges
fooddb_ex	ALTER, ALTER ROUTINE, CREATE, CREATE ROUTINE, CREATE TEMPORARY TABLES, CREATE VIEW, DELETE, DROP, EVENT, EXECUTE, INDEX, INSERT, LO...
practice	ALTER, ALTER ROUTINE, CREATE, CREATE ROUTINE, CREATE TEMPORARY TABLES, CREATE VIEW, DELETE, DROP, EVENT, EXECUTE, INDEX, INSERT, LO...

Privileges of the coms363 user.

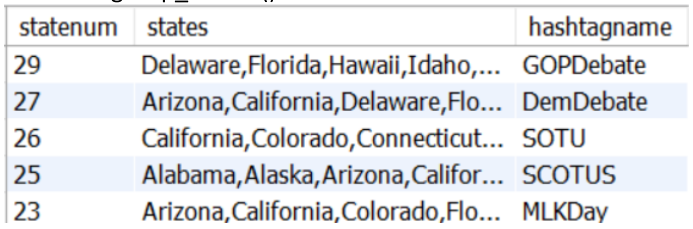
3. Create a new MySQL connection using 'coms363' as the user. For the above example, you see only two databases, fooddb_ex and practice. Create and run your stored procedure code as the coms363 user.

Part 1: Write stored procedures that will be called by your Java application.

You will need to look into the schemas and the data types of the attributes to use them for your stored procedure. Do not disable the foreign key check in your code or MySQL Workbench editor.

1. (25 points) Write a stored procedure named **findPopularHashTags** in a file named findPopularHashTags.sql. This stored procedure finds *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 in, and the hashtag itself in descending order of the number of states the hashtag appeared. The stored procedure accepts as the first parameter the value of k and the second parameter an integer that indicates the year of interest.

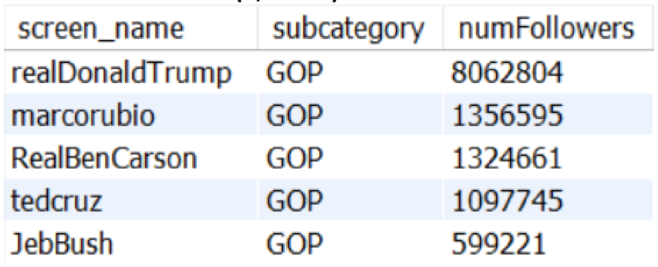
Output of
call findPopularHashTags(5, 2016)
Hint: Use group_concat() function to create a list of states



statenum	states	hashtagname
29	Delaware,Florida,Hawaii,Idaho,...	GOPDebate
27	Arizona,California,Delaware,Flo...	DemDebate
26	California,Colorado,Connecticut...	SOTU
25	Alabama,Alaska,Arizona,Califor...	SCOTUS
23	Arizona,California,Colorado,Flo...	MLKDay

2. (20 points) Write a stored procedure named **mostFollowedUsers** in a file named mostFollowedUsers.sql. This file accepts two parameters: k of type integer and name of the political party. The stored procedure finds *k* most followed users of 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. The users.numFollowers attribute stores the number of followers of this user. The user's party affiliation is stored in the users.subcategory attribute. The data type of the last parameter must be the same as the data type of the users.subcategory attribute.

Output of
call mostFollowUsers(5, 'GOP')



screen_name	subcategory	numFollowers
realDonaldTrump	GOP	8062804
marcorubio	GOP	1356595
RealBenCarson	GOP	1324661
tedcruz	GOP	1097745
JebBush	GOP	599221

3. (25 points) Write a stored procedure named **insertUser** in a file called insertUser.sql. This stored procedure inserts a new Twitter user into the users relation. The stored procedure accepts input parameters for all attributes of the users relation in the order of the attributes stored in the schema. The last parameter is an output-only parameter to indicate to the caller of the stored procedure whether the insertion is successful or not. A zero means false (a failed insertion), and a 1 means true (i.e., a successful insertion). The order of the parameter and the data type of each parameter need to be the same as those in the users relation.

Example test case:

```
set @screen_name="pakatisu";
set @user_name="pak at ISU";
set @category = null;
set @subcategory = "Independent";
set @state = "Iowa";
set @numFollows=0;
set @numFollowing=0;
-- 0 means false;
set @success = 0;
call insertUser(@screen_name, @user_name, @category, @subcategory, @state, @numFollows,
@numFollowing, @success);
select @success;
```

4. (30 points) Write a stored procedure named **deleteUser** in a file called deleteUser.sql. This stored procedure deletes a user from the users relation given the screen name of the user as an input parameter. The stored procedure also accepts the output parameter that returns -1 to indicate that the given user does not exist, 0 for a successful deletion, and 1 for a failed deletion.

Hint: A Twitter user may post tweets or mentioned in tweets posted by others; therefore, it is necessary to delete rows that reference this user explicitly or via tweets by this user before you can delete the user.

One test case to try:

```
set @errorcode = -2;
call deleteUser("ajc", @errorcode);
select @errorcode;
```

Note: It is a good idea to test your stored procedure well first before you call them in your Java code.

Part 2: Java database application development

Write a Java program that uses JDBC API to access the practice database.

The program asks for the database username and password. After a successful login, the program shows the main menu with the options *a-d* and *e*, waits for the user to choose one, and performs the corresponding task for that option. When the task is done, the program returns to the main menu and waits for the user input. The option *e* is for the user to terminate the program. See an example in the JDBCTransactionTester.java code on Canvas Week 8.

Option a (**15** points): Your Java code asks for the input of the stored procedure **findPopularHashTags,** calls the stored procedure, and prints the output.

Option b (15 points): Your Java code asks for the input of the stored procedure **mostFollowedUsers**, calls the stored procedure, and prints the output.

Option c (20 points): Your Java code asks for the input of the stored procedure **insertUser,** calls the stored procedure, and prints the result of the insertion, whether it is successful or not. You can determine the required input from the schema of the users relation. The program also asks for the values of the other non-required attributes. If your application user does not provide the values, set them to null to pass into the stored procedure.

Option d (25 points): Your Java code asks for the input of the stored procedure **deleteUser,** calls the stored procedure, and prints out the result of the deletion. The required input is the required attributes of the users relation.

Other point allocation (25 points)

- Seven points that the code has @author tag followed by the student's full name. The code has comments for each of your methods or long code sections.
- Seven points for the part of your Java code that (1) sets the connection string to use the 'coms363' user and the practice database and (2) shows the menu for the application user to see the menu options.
- Eleven points for calling setTransactionIsolation(Connection.TRANSACTION_SERIALIZABLE) and setAutoCommit, commit, and/or rollback in appropriate places to ensure that in the event that the application user aborts, your code does not create inconsistency to the database.

Other requirements:

- Name your Java class tweetsdb and put it inside the coms363 package. Add @author tag followed by your full name in the Java source file. If you use JDBCTransactionTester.java to build your code up on, add the @author tag after the comment about the ComS 363 Teaching Staff being the authors in the code. Add a comment indicating the change you made.
- Use the 'coms363' account for your database access. For the database connection string, set the host IP to 127.0.0.1 or localhost, which means the current host. Set the port number to 3306, which is the default port for MySQL. Use useSSL=true, which means your queries and results from DBMS are sent encrypted.
  In some systems, you may have to set other parameters, such as "allowPublicKeyRetrieval=true&useSSL=true" in your database connection string, to avoid the error regarding a public key.
- Your code must take advantage of MySQL to discard as many irrelevant rows as possible. Do not use JDBC API to retrieve irrelevant rows from the database only to discard them using your Java code. Such code will receive very low scores.
- Your code must use JDBC parameterized SQL statements to construct any queries that need the user's input as part of the queries. These statements are those with '?' where the value of the '? ' is set by another JDBC method call. Points will be deducted if you hard code the parameter values of the stored procedures.
- Make sure your code that involves insert, update, or delete SQL statements utilize transaction management provided by DBMS through JDBC calls: setTransactionIsolation(Connection.TRANSACTION_SERIALIZABLE), setAutoCommit, commit, and/or rollback in appropriate places. See an example in the JDBCTransactionTester.java.

- You are free to design your way of listing the returned result from the stored procedure in your Java code as long as all the requested attributes are shown.

Grading consideration

- Ensure all your stored procedure(s) work without setting the default database. For instance, use practice.tweets instead of tweets when you refer to the tweets table in the practice database.
- Your Java code must compile. We will test your code using Java version 13. The code that does not compile will initially receive a zero. To receive partial credit, contact the grading TA within one week after the grade of this assignment is released. The discussion is to resolve the compilation issue with the grading TA. After solving the compilation issues, partial credits will be given based on how well the code functions.
- We do not grade how nice the user interface looks as long as we can provide user input and easily see correct query results.
- For this assignment, you can write your code from scratch or adapt the instructor's provided code, JDBCTransactionTester.java, but do indicate that in the comment section of your code as described above.