COMS 363 Fall 2020
Assignment 4

Date assigned: Tuesday October 27, 2020
Due date: Thursday November 12, 2020 by 11:50pm
Percentage in your final grade: 9%
Maximum score for the assignment: 100 points

Learning objectives:

1. Practice writing a Java application using Java Database Connectivity (JDBC) for database access.
2. Protect against failure utilizing transaction support provided by DBMS via method calls: setAutoCommit, commit, and/or rollback.

Submission instruction: Name your Java file hw4.java. Submit this file. Do not submit the binary file.

IMPORTANT points

1. Add @author tag followed by your full name in the Java source file. Name your class hw4 and put it inside cs363 package.
2. Use the username cs363@% and the password "363F2020" for your database access. The @% means allowing the cs363 user to access from any hosts including remote hosts.
3. Set the host IP to 127.0.0.1 and port 3306 and useSSL=false in the database connection string. In some systems, you may have to set other parameters such as "allowPublicKeyRetrieval=true&useSSL=false" in your database connection string to avoid the error regarding a public key.
4. Your Java code must compile successfully in order to be graded. We will test your code using Java version 13. Java version 11 and higher should be ok with this assignment.
5. We do not grade on how nice the user interface looks as long as we can provide user input and see correct query results easily. You can adapt the instructor's provided code, JDBCTransactionTester.java for this assignment.
6. 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 Java code. Such code will receive very low scores.
7. Your code must use parameterized SQL statements to construct any queries that need the user's input as part of the queries.
8. Make sure that your code for each user's option that involves insert, update, or delete SQL statements fully utilizes MySQL to keep the database in the consistent state against failure and interference from other concurrent running programs.

   Hint: Recall the use of setAutocommit(false) and commit() methods. You can use the default JDBC transaction isolation level or higher.

Instruction

Write a Java program that uses JDBC API to access the sakila_mod database created for Homework 3. The program shows the main menu with the options *a-c* and *e*, waits for the user to choose one of the options, and performs the corresponding task. 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.

a. (30 points) Insert information about a new actor into the actor relation. The program asks for the first name and the last name of the new actor. It checks that the user's provided first name and the last name of the actor are not null. The program inserts a new row into the actor relation with the actor_id value set to the next highest integer of the existing actor_id values in the table. The values of the first_name and the last_name attributes of the row are set to the provided first name and last name, respectively. The value of the last_update attribute of the row is set to the value of the CURRENT_DATE() function.

Point allocation: 10 points for the Java code not relevant to the use of DBMS; 20 points for correct use of parameterized SQL statements, setAutoCommit() and the commit() method.

b. (35 points) Delete customer information given a customer id. The program asks for the customer id to be deleted. Before deleting the row in the customer relation, display a warning message that all the information related to this customer will be deleted. Ask the user "y" or "n" to proceed. If receiving "y", proceed to delete all the rows in other relations that reference this customer (i.e., payment and rental relations) and then delete this customer from the customer relation. Do not disable the foreign key check in your code or on your Workbench editor.

Point allocation: 10 points for the Java code not relevant to the use of DBMS; 25 points for correct use of parameterized SQL statements and the commit() method.

c. (25 points) Report the total sales for a given month. The program asks for the month (e.g., 2 for February) and uses a CallableStatement object to call the stored procedure my_total_sales given the user's specified month. The program shows the value of the total_amount parameter. Execute the following code to create the stored procedure first in the sakila_mod database.

Hint: Use methods of the CallableStatement class to set the values of the input parameters to the stored procedure and the registerOutParameter method to register the output parameter returned by the stored procedure.

Point allocation: 10 points for Java code for calling the stored procedure using parameterized SQL statement. 15 points for the code that shows the returned results.

```
/*****total_sales**********************************************************
DELIMITER &&
CREATE PROCEDURE my_total_sales(IN given_month INT, OUT total_amount double)
BEGIN
        SELECT sum(p.amount) INTO total_amount
  FROM payment p
  INNER JOIN rental AS r ON p.rental_id = r.rental_id
  WHERE month(r.rental_date)=given_month;
END &&
DELIMITER ;
```

The remaining 10 points are for other parts of Java code and comments.