DBMS LAB 09 TASKS

Prepared by:
Mohammad Anas Jawad
Lecturer, IUT CSE

Department of Computer Science and Engineering
Islamic University of Technology
August 1, 2021

Contents

1 SCENARIO

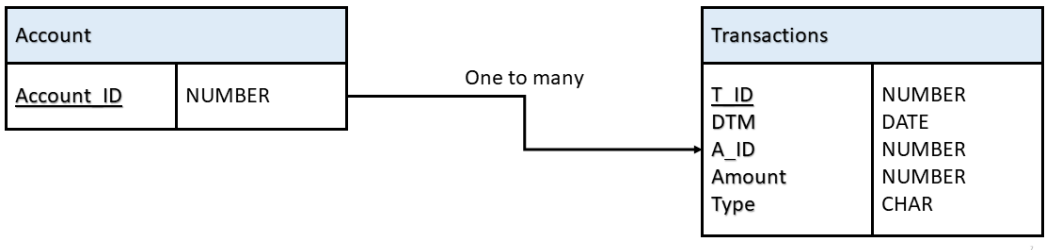

Figure 1: Given Schema

For simplicity, ACCOUNT table contains only the account ID. TRANSACTIONS table contain the information related to each transaction that occurs in the bank. The table stores transaction ID, date of the transaction, account involved in the transaction, the amount transacted. The TYPE column stores 0 if the money is transferred to the account, that is, the money is added to the account. It stores 1 if the money is transferred from the account, that is, money is subtracted from the account.
There are 3 types of accounts in the bank.

- If the user has a total final balance of more than 1000000 and the total amount of transactions (both debit and credit) the person has made totals more than 5000000, that person is regarded as Commercially Important Person (CIP).
- If the user has a total final balance that is more than 500000 but less than 900000 and the total amount of transactions the person has made totals more than 3500000 but less than 4500000, that person is regarded as Very Important Person (VIP).
- If the user has a total final balance that is less than 40000 and the total amount of transactions the person has made totals less than 3000000, that person is regarded as Ordinary.

2 TASKS AND INSTRUCTIONS

Carry out the following instructions:

1. Download and execute `TableGenerator.cpp`. It will create a file named `table.sql` which contains the required SQL queries to create and populate the database. The values will be random except for the account numbers. (These have been simplified for later purposes).

2. Make sure Oracle Database is installed in your PC. Login to your database (either using Command Prompt or SQL Command Line). Enter the command: `@directory\table.sql`. Here, (directory) should be replaced by the file path of table.sql file. If the file is stored in D: drive, the command will look like:

   ```
   @D:\table.sql
   ```

3. Create a new Java project and add ojdbc8.jar or ojdbc10.jar file as an external JAR file.

4. Go through section 3 to take a look at a sample solution. You can use it as a reference for your solution. [Additional hint: loops and if-else blocks in java have the same syntax as in C++.]

5. Now, write a JAVA code to:

   - Establish connection to the database.
   - Fetch required data from the `TRANSACTIONS` table.
   - Count the total number of accounts, number of CIP, VIP and Ordinary persons.
   - Also count the number of persons that were uncategorized.

     The output of your program should look like the following:

```
Total Number of Accounts: 147
CIP count: 25
VIP count: 2
Ordinary count: 1
Uncategorized: 119
```

Note that the numbers will not be the same for you. As your table entries will be generated randomly, everyone should have different answers. If I find any output with similar values, you can already guess what will happen.

6. Add a comment above each variable declaration, explaining what it stores.

7. Add comments above each function call (if any), explaining what it does and what parameters are passed to the function.

8. Create a zip file containing your JAVA code, your `table.sql` file and and output.txt file that contains your output like - the total number of accounts, total CIP, VIP, Ordinary and Uncategorized count. You don't have to generate the output.txt file using your program, simply copy your program's output to the text file.
   Rename the zip file using the format `Lab09_STUDENT_ID.zip` and submit it via Google Classroom. Example: For student with ID 154443, the file name will be `Lab09_154443.zip`

9. **NOTE:** Some of you may face issues where your output does not generate any count for CIP or VIP. In such cases, check your generated table.sql file and see what the maximum amount of transaction is in your TRANSACTIONS table. If it is something around 140,000, it probably means your compiler cannot generate larger random numbers. If you check the cpp file, you will see that I have set the upper limit for the transaction amount to be generated to 1,000,000; so 140,000 should not be the maximum amount in your table.
   Anyway, if that's the case, I advise using any online compiler like `www.onlinegdb.com` to execute the cpp file and then use the generated table.sql file. For reference, check

the RAND_MAX value generated by your compiler. It should be aound 2147483647. If it's less than that, you will probably have trouble generating CIP and VIP counts.

3 SAMPLE SOLUTION

SampleSolution.java

```java
import java.sql.*;
public class SampleSolution {
    public static void main(String[] args) {
        String username = "senpai";
        String password = "senpai";
        String url = "jdbc:oracle:thin:@localhost:1521/XE";
        String sqlQuery = "SELECT A_ID, AMOUNT, TYPE FROM TRANSACTIONS";

        int[] balance = new int[1000];
        int[] total = new int[1000];
        int account;
        int amount;
        String type; /* Hint: These variables will help you with your code. You might also need some other variables though! */
        try {
            // 1) Register the driver class
            Class.forName("oracle.jdbc.driver.OracleDriver");
            // 2) Create the connection object
            Connection con = DriverManager.getConnection(url, username, password);
            // 3) Create the Statement object
            Statement statement = con.createStatement();
            System.out.println("Connection to database successful");
            // 4) Execute the query
            ResultSet result = statement.executeQuery(sqlQuery);
            while (result.next()) {
                account = result.getInt("a_id"); //Could also be
```

```
    written as result.getInt(1);
26                  amount = result.getInt("amount"); //Could also be
    written as result.getInt(2);
27                  type = result.getString("type"); //Could also be
    written as result.getInt(3);
28                  System.out.println(account + " " + amount + " " + type)
    ;
29                  /* Additional Code */
30              }
31              /* Additional Code */
32              // 5) Close the connection object
33              con.close();
34              statement.close();
35              result.close();
36          } catch (SQLException e) {
37              System.out.println("Error while connecting to database.
    Exception code: " + e);
38          } catch (ClassNotFoundException e) {
39              System.out.println("Failed to register driver. Exception
    code: " + e);
40          }
41          System.out.println("Thank You!");
42      }
43  }
```