Database Systems Project 2 - Using PL/SQL and JDBC to Implement the Retail Business Management System

(Due: 10pm, April 20, 2023)

This project is to use Oracle's PL/SQL and JDBC to implement the RBMS application. This is a team project. Each team must have at least two students and at most three students. Teams are formed by students themselves. A team may consist of students from all database sections (CS432 and CS532). You can post messages about forming teams in the Project 2 thread of the discussion forum in Brightspace.

Due to time constraint, only a subset of the database tables and a subset of the desired functionalities will be implemented in this project.

1. Preparation

Due to the limited time, we will use only the following six tables for this project:

- Employees(eid, name, telephone#, email)
- Customers(cid, first_name, last_name, phone#, visits_made, last_visit_date)
- Products(pid, name, qoh, qoh_threshold, orig_price, discnt_category)
- Prod_Discnt(discnt_category, discnt_rate)
- Purchases(pur#, eid, pid, cid, pur_time, quantity, unit_price, payment, saving)
- Logs(log#, user_name, operation, op_time, table_name, tuple_pkey)

The first five tables are the same as the tables used in Project 1. Each tuple in the logs table describes who (the login name of a database user) has performed what operation (insert, delete, update) on which table (give the table name) and which tuple (as indicated by the value of the primary key of the tuple) at what time (date and time). Attribute log# is the primary key of this table.

The scripts for tables Employees, Customers, Products, Prod_Discnt and Purchases have been given in Project 1. The script for creating the logs table is given below:

```
create table logs
(log# number(4) primary key,
user_name varchar2(12) not null,
operation varchar2(6) not null,
op_time date not null,
table_name varchar2(20) not null,
tuple_pkey varchar2(6));
```

You don’t need to populate the logs table manually with any tuples. This table will be populated later by triggers.

2. PL/SQL Implementation (58 points)

You need to create a PL/SQL package for this application. All procedures and functions should be included in this package. It is suggested that you create other Oracle objects such as sequences and triggers outside the package. The following requirements and functionalities need to be implemented.

1. (4 points) Create two sequences – one will be used to automatically generate unique values for pur# and the other will be used to automatically generate unique values for log#. The sequence for pur# needs to generate numbers that start with 10001 and the sequence for log# needs to generate numbers that start with 1001.

   You can delete all tuples from table Purchases and re-populate this table using the same tuples except that pur# will be generated by values using the sequence for pur# like the examples below:

   ```
   insert into purchases values (seqpur#.nextval, 'e01', 'p002',
   'c001', to_date('12-AUG-2022', 'DD-MON-YYYY'), 1, 211.65, 211.65,
   37.35);
   insert into purchases values (seqpur#.nextval, 'e01', 'p003',
   'c001', to_date('20-DEC-2022', 'DD-MON-YYYY'), 1, 118.40, 118.40,
   29.60);
   ```

2. (5 points) Create a procedure to show the tuples in each of the six tables for this project. If you just want to run your package in the SQL*Plus environment, it is sufficient to create a procedure for each table. As an example, you can implement a procedure, say **show_employees**, in your package to show all employees in the employees table. On the other hand, if you want to get the results to your JDBC interface program, you should create a function and use *ref cursor* (see sample program 3 for the use of *ref cursor*).

3. (4 points) Create a procedure to report the monthly sale activities for any given employee. For example, you can use a procedure, say **monthly_sale_activities(employee_id)**, for this operation. For the given employee id (an in parameter), you need to report the employee id, employee name, the month (the first three letters of the month, e.g., MAR for March), the year (4 digits), the total number of times the employee made sales (i.e., the number of purchases that involve the employee) each month, the total quantity sold by the employee each month, and the total dollar amount of the sales made by the employee each month. Only need to list the information for those months during which the given employee has actually made sales.

4. (8 points) Create a procedure for adding tuples to the Employees table. You may use a procedure, say **add_employee(e_id, e_name, e_telephone#, e_email)**, in your package to add a tuple into the Employees table, where e_id, e_name, e_telephone# and e_email are all in parameters of the procedure.

   Create a trigger that can add a tuple to the logs table automatically whenever a new employee is added to the Employees table. For the tuple to be added to the logs table in this case, the

table_name is “employees”, the operation is “insert” and the tuple_pkey is the eid of the newly inserted employee.

5. (22 points) Create a procedure for adding tuples to the Purchases table. You may use a procedure, say **add_purchase(e_id, p_id, c_id, pur_qty, pur_unit_price)**, in your package for this purpose, where e_id, p_id, c_id, pur_qty and pur_unit_price are all in parameters of the procedure. Note that the rest of the attribute values of the new purchase tuple should be automatically generated. Specifically, *pur#* should be generated by your sequence, *payment* should be computed based on pur_qty and pur_unit_price, *saving* should be computed based on pur_qty, pur_unit_price as well as the orig_price of the product from the Products table, and pur_time should be generated by sysdate.

   Before a tuple is added into the Purchases table, your procedure needs to make sure that, for the involved product, the quantity to be purchased is equal to or smaller than the quantity on hand (qoh). Otherwise, an appropriate message should be displayed (e.g., “Insufficient quantity in stock.”) and the purchase request should be rejected.

   After adding a tuple to the Purchases table, the qoh attribute of the Products table should be modified accordingly, that is, the qoh of the product involved in the purchase should be reduced by the quantity purchased. If the purchase causes the qoh of the product to be below qoh_threshold, your procedure should perform the following tasks:

   (a) Print a message “The current qoh of the product is below the required threshold and new supply is required.”
   (b) Automatically order supply for the product. This will be simulated by resetting the value of qoh to be qoh_threshold + 20.
   (c) Print another message stating the new value of the qoh of the product after new supply.

   In addition, the insertion of the new tuple into the Purchases table will increase the visits_made of the involved customer by one and the last_visit_date needs to be updated if the purchase date is different from the last_visit_date.

   (10 of the 22 points are for the triggers) Create triggers to implement the update of qoh, the printing (displaying) of the messages, and the updates of visits_made and last_visit_date.

6. (10 points) Create triggers that can add tuples to the logs table automatically whenever the following events happen: (1) update the last_visit_date attribute of the Customers table; (2) update the visits_made attribute of the Customers table; (3) insert a tuple into the Purchases table; and (4) update the qoh attribute of the Products table. When a tuple is added to the logs table due to the first or the second event, the table_name is “customers”, the operation is “update” and the tuple_pkey is the cid of the affected customer. When a tuple is added to the logs table due to the third event, the table_name is “purchases”, the operation is “insert” and the tuple_pkey should be the pur# of the newly inserted purchase. When a tuple is added to the logs table due to the fourth event, the table_name is “products”, the operation is “update” and the tuple_pkey is the pid of the affected product. You need to implement four triggers for this task, one for each event.

7. (5 points) You need to make your package user friendly by designing and displaying appropriate messages for all exceptions. For example, if someone wants to find the purchases of a customer but entered a non-existent customer id, your program should report the problem clearly.

3. Interface (30 points)

Implement an interactive and menu-driven interface using Java and JDBC (see sample programs). Your interface program should utilize as many of your PL/SQL code as possible. Note that messages that are printed by the dbms_output.put_line( ) in your PL/SQL package or triggers are not visible when you run your Java/JDBC application. You may use dbms_output.get_line( ) to retrieve messages that were place into the buffer by dbms_output.put_line( ) and call dbms_output.get_line( ) from your Java program. Alternative, it is acceptable to regenerate these messages in your Java program.

4. Documentation (12 points)

Documentation consists of the following aspects:

1. Each procedure and function and every other object you create for your project needs to be explained clearly regarding its objective and usage.

2. Your code needs to be well documented with in-line comments.

3. *Team report*. This report should describe in reasonable detail how the team members collaborated for the project. More specifically, it needs to include the following information:
    a. Your meetings – describe when each meeting occurred and what was discussed for the project at the meeting.
    b. Your plans – describe your team's plan (schedule) for completing your project. Also report how the plan was followed during the course of completing the project.
    c. Your responsibilities – describe how the tasks are divided among the team members. Specifically, describe which member is primarily responsible for which part of the project and how the other members contribute to the task primarily assigned to a particular member.
    d. Your self-assessment of the team work – which of the following phrases best describe how well your team has worked together: (1) worked really well together; (2) generally worked well with some minor issues; (3) struggled as a team but made it work eventually; (4) struggled throughout mostly. Your self-assessment will NOT affect the grade of your project.

4. Each team member may also submit a separate personal report to the instructor by email if you have disagreement with the team report. In the personal report, describe clearly how you disagree with the team report. If you completely agree with the team report, you don't need to submit a personal report. Personal reports will be kept confidential by the instructor.

5. Hand-ins, Demo and Grading

1. Each team needs to submit a report containing the following components (only one copy for each team is needed and one member can submit it on behalf of the team) to the Project 2 submission folder:

    - Names of team members.

    - The honesty statement: "We have done this assignment completely on our own. We have not copied it, nor have we given our solution to anyone else. We understand that if we are involved in plagiarism or cheating we will have to sign an official form that we have cheated and that this form will be stored in our official university records. We also understand that we will receive a grade of 0 for the involved assignment and our grades will be reduced by one level (e.g., from A to A- or from B+ to B) for our first offense, and that we will receive a grade of "F" for the course for any additional offense of any kind."

    - The *Team Report* (see the Documentation section).

    - The entire PL/SQL code (including the package, triggers, and sequences).

    - The entire Java code.

2. If needed, a student may submit a personal report to the instructor by email.

3. Each team is required to demonstrate the completed project to the instructor using tuples created by the instructor. More instructions on demo will be provided before the demo.

4. The grading will be based on the quality of your code, the documentation and on how successful of your demo is.