CS432 Project 2 - Using PL/SQL and JDBC to Implement the Retail Business Management System

(Due: November 27, 2017)

This project is to use Oracle's PL/SQL and JDBC to implement the RBMS application. **Up to three students may work together as a group** for this project. A team may consist of students from all database sections. 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 (5 points)

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

Employees(eid, name, telephone#, email)
Customers(cid, name, telephone#, visits_made, last_visit_date)
Products(pid, name, qoh, qoh_threshold, original_price, discnt_category)
Discounts(discnt_category, discnt_rate)
Suppliers(sid, name, city, telephone#, email)
Supplies(sup#, pid, sid, sdate, quantity)
Purchases(pur#, eid, pid, cid, ptime, qty, total_price)

In addition, the following table is also needed for this project:

Logs(log#, user_name, operation, op_time, table_name, tuple_pkey)

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 and Purchases have been given in Project 1. The scripts for creating the remaining tables are given below (please note that table Products has been revised from Project 1). If you need to recreate a table that already exists, you need to drop the table first. Please also note that the tables are created in certain order such that by the time when a foreign key needs to be created, the table containing the corresponding primary key has already been created.

```
create table discounts
(discnt_category number(1) primary key check(discnt_category in (1, 2, 3, 4)),
discnt_rate number(3,2) check (discnt_rate between 0 and 0.8));

create table products
(pid char(4) primary key,
name varchar2(15),
qoh number(5),
qoh_threshold number(4),
original_price number(6,2),
discnt_category number(1) references discounts);
```

```
create table suppliers
(sid char(2) primary key,
name varchar2(15) not null unique,
city varchar2(15),
telephone# char(12) not null unique,
email varchar2(20) unique);

create table supplies
(sup# number(4) primary key,
pid char(4) references products(pid),
sid char(2) references suppliers(sid),
sdate date,
quantity number(5),
unique(pid, sid, sdate));

create table logs
(log# number(5) 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 should populate the tables (except the Logs table) with appropriate tuples to test your program.

2. PL/SQL Implementation (55 points)

You need to create a PL/SQL package for this application. All procedures and functions should be included in this package. Other Oracle objects such as sequences and triggers are to be created outside the package. The following requirements and functionalities need to be implemented.

1. (3 points) Create a sequence to automatically generate the values for each of the three attributes pur#, sup# and log# when new tuples are inserted into the corresponding table. For each of these primary key attributes, if its data type is number(n), its values should have n digits (this can be achieved by starting a sequence with a value of n digits). Implement a different sequence for each of three attributes.

2. (7 points) Create a procedure or function to show the tuples in each table. 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. You need to implement 8 procedures or functions, one for each table. Alternatively, 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. (3 points) Create a function, say **purchase_saving(pur#)**, to report the total saving of any purchase for any given pur# (as an in parameter).

4. (4 points) Create a procedure to report the monthly sale activity information 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., FEB for February), 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 sold by the employee each month. Only need to list the information for those months during which the given employee has actual made sales.

5. (3 points) Create a procedure for adding tuples to the Customers table. You may use a procedure, say **add_customer(c_id, c_name, c_telephone#)**, in your package to add a tuple into the Customer table, where c_id, c_name, and c_telephone# are all in parameters of the procedure. Note that the initial values of visits_made and last_visit_date of any newly added customer should be generated by your procedure. Specifically, visits_made should be given an initial value of 1 and the initial value for last_visit_date can be generated by sysdate.

6. (10 points) Create triggers that can add tuples to the logs table automatically whenever certain events happen. In this project, the following events need to be tracked: (1) insert a tuple into the Customers table; (2) update the last_visit_date attribute of the Customers table; (3) insert a tuple into the Purchases table; (4) update the qoh attribute of the Products table; and (5) insert a tuple into the Supplies table. When a tuple is added to the logs table due to the first event, the table_name should be “customers”, the operation should be “insert” and the tuple_pkey should be the cid of the newly inserted customer. When a tuple is added to the logs table due to the second event, the table_name should be “customers”, the operation should be “update” and the tuple_pkey should be the cid of the affected customer. When a tuple is added to the logs table due to the third event, the table_name should be “purchases”, the operation should be “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 should be “products”, the operation should be “update” and the tuple_pkey should be the pid of the affected product. When a tuple is added to the logs table due to the fifth event, the table_name should be “supplies”, the operation should be “insert” and the tuple_key should be the sup# of the newly inserted supply. You need to implement five triggers for this task, one for each event.

7. (20 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)**, in your package for this purpose, where e_id, p_id, c_id and pur_qty are all in parameters of the procedure. Note that the pur# of any newly added purchase should be automatically generated by your sequence. In addition, total_price should be computed based on the data in the database automatically and ptime should be automatically generated by sysdate.

   Before a tuple is added into the 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 column 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 saying that the current qoh of the product is below the required threshold and new supply is required
(b) Automatically order supply for the product (i.e., add a new tuple to the Supplies table): the sup# is automatically generated by a sequence, the pid of the new supply is the pid of the product involved in the purchase, the sid of the new supply is the sid of a supplier who has supplied this product before (there should be such information in the current Supplies table; if multiple suppliers have supplied this product before, use the supplier with the smallest sid), the quantity of the new supply should be computed using 10 + M + qoh, where M is the minimum value for quantity such that M + qoh > qoh_threshold, and use sysdate for sdate,
(c) Increase qoh of the product by the quantity ordered.
(d) Print another message showing the new value of the qoh of the product.

In addition, the insertion of the new tuple into the Purchases table may cause the visits_made of the involved customer to be increased by one if the purchase is made on a new date and the last_visit_date may also have updated accordingly.

Use triggers to implement the update of qoh, the printing (displaying) of the messages, the insertion of new supplies to the Supplies table, and the updates of visits_made and last_visit_date.

8. (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 package in your PL/SQL package or triggers are not visible when you run your Java/JDBC application. You may need to regenerate these messages in your Java program.

A nice GUI or Web interface will receive a 10-point bonus.

4. Documentation (10 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. If your project team has more than one member, your team needs to submit a “Team Report”. This report should describe in reasonable detail how your team worked together for the project, including information such as who is primarily responsible for which part(s) of the project, how many times your team has met to discuss the project, what is the experience for each of you and what are the lessons learned.

4. Each team member also needs to submit a separate personal report about your team activities from your perspective. This report can be as simple as a single sentence if you agree with the team report (just say that you agree with the team report). You can also use this report to detail your disagreement with the team report or simply provide additional information about the team activities beyond the team report. Personal reports are submitted separately from other project documents and will be kept confidential by the instructor.

5. Hand-ins, Demo and Grading

1. Each team needs to hand in ONE hard copy of your entire PL/SQL code (including the package, triggers, and sequences). Only one copy for each team is needed.

2. Each team needs to submit all source code, including both PL/SQL and Java, to blackboard (Project 2 submission folder). Make sure to include information about the team members.

3. Each team needs to submit a team report in hard copy.

4. Each student working in a team needs to submit a hard copy of your personal report.

5. 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.

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

All reports must be typed and printed and submitted to the instructor by the specified time.