

CS532 Project 2 - Using PL/SQL and JDBC to Implement the Retail Business Management System

(Due: May 5, 2008 (Monday))

This project is to use Oracle's PL/SQL and JDBC to implement the RBMS application. Up to two students may work together as a group for this project. Due to time constraint, only a subset of the database tables and a subset of the needed 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 six tables in this project. In addition, we make the following changes to some tables to make the implementation simpler: (1) remove attribute `d_start_date` from the products table (implying that the discount rate does not change over time), (2) remove attribute description from the product table, and (3) add attribute `total_price` to the purchases table.

Employees2(eid, ename, city, telephone#)

Products2(pid, pname, qoh, qoh_threshold, orig_price, disc_category, sid)

Product_discount2(disc_category, disc_rate)

Suppliers2(sid, sname, city, telephone#)

Customers2(cid, cname, telephone#, visits_made)

Purchases2(pur#, eid, pid, cid, qty, total_price, pdate)

In addition, the following table is required for this project:

Logs(who, time, what)

Each tuple in the logs table describes who (the login name of a database user) has made what change (insert into and update of what table) at what time (date).

Please use the following SQL DDL statements to create the seven tables required for this project. Note that you need to use the exact statements as shown below to ensure that the instructor can test your programs using the instructor's data later. 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 corresponding primary key has already been created.

```
create table employees2
(eid integer primary key,
ename varchar2(10) not null,
city varchar2(15),
telephone# char(12));
```

```
create table product_discount2
(disc_category number(1) primary key check (disc_category in (1, 2, 3, 4)),
disc_rate number(3,2) check (disc_rate <= 1.0));
```

```

create table suppliers2
(sid integer primary key,
sname varchar2(15) not null unique,
city varchar2(15),
telephone# char(12));

create table products2
(pid integer primary key,
pname varchar2(10),
qoh number(5),
qoh_threshold number(5),
orig_price number(6,2),
disc_category number(1) references product_discount2,
sid integer references suppliers2);

create table customers2
(cid integer primary key,
cname varchar2(10) not null,
telephone# char(12),
visits_made number);

create table purchases2
(pur# integer primary key,
eid integer references employees2,
pid integer references products2,
cid integer references customers2,
qty number(6),
total_price number(6,2),
pdate date);

create table logs
(who varchar2(10) not null,
time date not null,
what varchar2(40) not null);

```

The meanings of most of the tables and their attributes are clear. If they are not clear to you, please let the instructor know. For a given customer, `visits_made` indicates how many times the customer has purchased products from the business. To simplify, all purchases made on the same day by the same customer are considered to be made in the same visit. This means that multiple visits made on the same day by the same customers are counted as one visit. Attribute `qoh` in the `products2` table indicates quantity on hand and for each product, `qoh_threshold` is an integer such that when `qoh` becomes less than `qoh_threshold`, it is time to get new supplies of this product from the suppliers. Each tuple in the `purchases2` table tells which customer (`cid`) has purchased what product (`pid`) with what quantity (`qty`) and the total price (`total_price`) from which employee (`eid`) at what date (`pdate`). The total price is computed by multiplying the discount price of the product and the quantity purchased. The discount price of a product is computed based on its original price and its discount

rate, which in turn is determined by `disc_category`. Each tuple in the table logs describes who (the login name of a database user) has made what change (insert into and update of what table) at what time (date).

You should populate the above tables with appropriate tuples to test your program.

2. PL/SQL Implementation (55 points)

You need to create a PL/SQL package and possibly other Oracle objects (sequences, triggers, ...) to manage the business. The following requirements and functionalities need to be implemented.

1. (3 points) All id (`pur#`, `eid`, `pid`, `cid`, `sid`) values are generated by appropriate sequences automatically when you add new tuples. All these id values should have four digits starting with 1000. Implement a different sequence for each of these id attributes.
2. (5 points) Can display the tuples in each table. As an example, you can implement a procedure, say **`show_products`**, in your package to display all products in the `products2` table. You need to implement seven procedures, one for each table.
3. (5 points) Can report the monthly sale information for any given product. For example, you can use a procedure, say **`report_monthly_sale(prod_id)`**, for this operation. For the given product id, you need to report the product name, the month (the first three letters of the month, e.g., FEB for February), the total quantity sold each month, the total dollar amount sold each month, and the average sale price (the total dollar amount divided by the total quantity) of each month. Only need to list the information for those months during which the given product has been purchased by some customers.
4. (7 points) Can add tuples to all tables except the logs table. In this project, you are only required to implement procedures to add tuples into the `purchases2` table and the `employees2` table. As an example, you can use a procedure, say **`add_purchase(e_id, p_id, c_id, pur_qty)`**, in your package to add a tuple in the `purchases2` table, where `e_id`, `p_id`, `c_id` and `pur_qty` are parameters of the procedure. Note that for this example, 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 `pdate` should be the current date (use `sysdate`).
5. (10 points) Can add a tuple to the logs table automatically whenever any table is modified. To simplify, you are only required to consider the following two modifications (events): insert a tuple into the `purchases2` table; update the `disc_category` attribute of the `products2` table. When a tuple is added to the logs table due to the first event, the value under attribute “what” should be “A new purchase with `pur#` xxxx is added to the `purchases2` table.”, where xxxx is the 4-digit `pur#` of the newly added purchase tuple. When a tuple is added to the logs table due to the second event, the value under attribute “what” should be “The discount category of product xxxx is changed from Y to Z.”, where xxxx is the 4-digit `pid` of the affected product, Y and Z are the old and new discount categories, respectively. Adding tuples to the logs table should be

implemented using triggers. You need to implement two triggers for this task, one for each event.

6. (5 points) Before a purchase is actually made (i.e., before a tuple is added into the purchases2 table), your program 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 should be rejected.
7. (15 points) After adding a tuple to the purchases2 table, the qoh column of the products2 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 program should print a message indicating (a) the current quantity on hand and (b) a new order of the product needs to be placed. In addition, the insertion of the new tuple in the purchases2 table may cause the visits_made of the customer to be increased by one if the purchase is made on a new date. Use triggers to implement the update of qoh, the sending of the message and the update of visits_made.
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 in the bingsuns environment using Java and JDBS (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 may not be visible when you run your Java/JDBC application. You may need to regenerate these messages in your Java program.

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. Relationships among different objects (including procedures and functions) need to be made clear (a diagram can be used).
2. Your code needs to be well documented with in-line comments.
3. If you work with another student as a team, you need to describe in detail how you two collaborated for the project, who is primarily responsible for which part, what is the experience for each of you and what are the lessons learned.

5. Hand-ins, Demo and Grading

1. You need to hand in a hardcopy of your entire code together with your documentation for the project. Only one copy for each team is needed.
2. All teams are required to demonstrate your project to the instructor using tuples created by the instructor. More instructions on demo will be given before the demo.
3. The grading will be based on the quality of your code, the documentation and on how successful of your demo is.