CS-532-02 Project 2 Xiaoyang Zhang

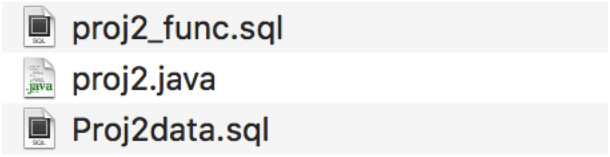
I have done this assignment completely on my own. I have not copied it, nor have I given my solution to anyone else. I understand that if I am involved in plagiarism or cheating I will have to sign an official form that I have cheated and that this form will be stored in my official university record.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

Signature:

One person team: Xiaoyang Zhang

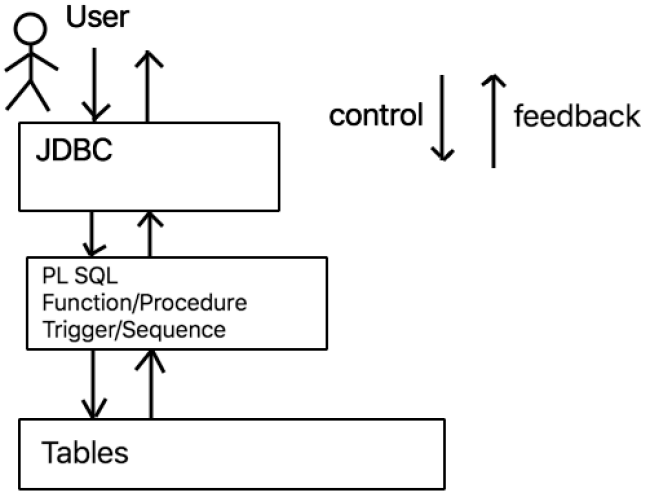
Summary:
Retail Business Management System
Language: PL SQL, Java(JDBC)
Running environment: terminal, bingsuns server.
UI: terminal UI, not web UI or GUI.

File Structure:



A. proj2.java
   This code contains UI, JDBC interface, procedure control.
B. proj2_func.sql
   This code contains all the function and procedure blocks which this project required.
C. Proj2data.sql
   C_1: This code contains three sequence trigger generator (pur#, sup#, log#).
   C_2: This code contains other triggers besides three sequence triggers.
   C_3: All the tables which are needed in project2 are created in this code.
   C_4: Initialize the tables, inserts tuples in the tables.

Code Structure:
Three levels:



Detail information:
A: proj2.java

1. Connector to server
   It is safe to read key from keyboard.

```
readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
System.out.print("Please enter user name >>> ");
userbuffer = readKeyBoard.readLine();
System.out.print("Please enter keyword >>> ");
keybuffer = readKeyBoard.readLine();

//Connecting to Oracle server. Need to replace username and
//password by your username and your password. For security
//consideration, it's better to read them in from keyboard.
OracleDataSource ds = new oracle.jdbc.pool.OracleDataSource();
ds.setURL("jdbc:oracle:thin:@castor.cc.binghamton.edu:1521:acad111");
Connection conn = ds.getConnection(userbuffer, keybuffer);
```

2. Connector to PL SQL and PL SQL Instruction input
   In JDBC, string is used for keeping the instruction which is used in PL SQL, such as bufferString in the below code.

```
if(commandName.equals("save")){
  buffer = 9;
  bufferString = "begin ? := proj2.purchase_saving("+ showPur +"); end;";
  commandName = null;
}
else if(commandName.equals("activity")){
  buffer = 10;
  bufferString = "begin proj2.monthly_sale_activities('"+ inputEID +"',:1); end;";
  commandName = null;
}
```

   Then we use prepareCall method to send instruction to PL SQL to call the function or procedure in PL SQL.

```
CallableStatement cs = conn.prepareCall(bufferString);
```

   Sometimes the SQL function or procedure has output parameters, we can use registerOutParameter method to set the type of these parameters.

```
cs.registerOutParameter(1, OracleTypes.INTEGER);
cs.registerOutParameter(2, OracleTypes.INTEGER);
```

3. Procedure control
   In the Java part, I use switch…case and if…else to control the program procedure.
   First, read the command from keyboard.
   Second, base on the input command, get more parameters from keyboard.

```
else if(commandName.equals("save")){
  System.out.print("Please enter pur# number: ");
  showPur = readKeyBoard.readLine();
}else if(commandName.equals("activity")){
  System.out.print("Please enter employee_ID: ");
  inputEID = readKeyBoard.readLine();
}else if(commandName.equals("+customer")){
  System.out.print("Please enter customer cid: ");
  inputCustomersCID = readKeyBoard.readLine();
  System.out.print("Please enter customer name: ");
  inputCustomersName = readKeyBoard.readLine();
  System.out.print("Please enter customer tel: ");
  inputCustomersTel = readKeyBoard.readLine();
}else if(commandName.equals("+purchase")){
  System.out.print("Please enter purchase eid: ");
  inputPurchasesEID = readKeyBoard.readLine();
  System.out.print("Please enter purchase pid: ");
  inputPurchasesPID = readKeyBoard.readLine();
  System.out.print("Please enter purchase cid: ");
  inputPurchasesCID = readKeyBoard.readLine();
  System.out.print("Please enter purchase qty: ");
  inputPurchasesQTY = readKeyBoard.readLine();
```

Third, base on the first and second steps, the PL SQL instruction can be generated. In this step, a buffer value is assigned to each instruction.

```
//System.out.println("commandName >>> " + commandName);
if(commandName.equals("save")){
  buffer = 9;
  bufferString = "begin ? := proj2.purchase_saving("+ showPur +"); end;";
  commandName = null;
}
else if(commandName.equals("activity")){
  buffer = 10;
  bufferString = "begin proj2.monthly_sale_activities('"+ inputEID +"',:1); end;";
  commandName = null;
}
else if(commandName.equals("+customer")){
  buffer = 11;
  bufferString = "begin proj2.add_customer('"+ inputCustomersCID +"','"+ inputCustomersName +"','"+
      inputCustomersTel +"'); end;";
  commandName = null;
}
```

Last step, base on the buffer value, related operation can be run correctly.

```
switch(buffer){
      case 1:
            System.out.print("\n");
            formatter.format("%-3s %-15s %-12s %-20s\n", "EID", "NAME", "TELEPHONE#", "EMAIL");
            while (rs.next()) {
            //System.out.println("-------------------------------------------------");
            //System.out.println(rs.getString(1) + "\t" +
            //rs.getString(2) + "\t" + rs.getString(3) + "\t" + rs.getString(4));
            formatter.format("-------------------------------------------------\n");
            formatter.format("%-3s %-15s %-12s %-20s\n", rs.getString(1), rs.getString(2),
                rs.getString(3), rs.getString(4));
            }
            System.out.print("\n");
      break;
```

4. Information output
   There are two ways to output data in Java part. One is output a table, ref-cursor should be used in this way.

```
case 2:
      System.out.print("\n");
      formatter.format("%-4s %-15s %-12s %-4s  %-12s\n", "CID", "NAME", "TELEPHONE#", "VISITS_MADE",
          "LAST_VISIT_DATE");
      while (rs.next()) {
      formatter.format("-------------------------------------------------------------------\n");
      formatter.format("%-4s %-15s %-12s %-12s %-12s\n", rs.getString(1), rs.getString(2),
          rs.getString(3), rs.getString(4), rs.getString(5));
      }
      System.out.print("\n");
```

Another way is output a sentence. Before that we should get some return data from PL SQL function or output value from procedure. In below case, getInt(1) is the return value from PL SQL.

```
if(buffer == 12 && cs.getInt(1) == 0){
  System.out.println("Insufficient quantity in stock.\n");
}else if(buffer == 12 && cs.getInt(1) == 2){
  System.out.println("Current qoh of the product is below the required threshold.");
  System.out.println("New supply is reguired, make new order.");
  System.out.println("Product ID >> " + inputPurchasesPID + "   update quantity of hold >> " + cs.getInt(2));
}
```

5. Catch Exception

   Most of time, we run the program in java environment. If we want to get some exception information from PL SQL part, we should catch the SQLException.

```
public static void main (String args []) throws SQLException {
 try
 {
```

Program body should be included in Exception catch mechanism.

```
    catch (SQLException ex) { System.out.println ("\n*** SQLException caught ***\n" + ex.getMessage());}
    catch (Exception e) {System.out.println ("\n*** other Exception caught ***\n");}

   }//main
 }//class
```

B: proj2_func.sql

All the PL SQL functions and procedures are included in this file.

1. There 8 show table functions in this code. Each of them use ref-cursor to return the table to JDBC.

```
function show_employees
return ref_cs_emp;

function show_customers
    return ref_cs_cus;

function show_discounts
    return ref_cs_dis;

function show_products
    return ref_cs_pro;

function show_purchases
    return ref_cs_pur;

function show_suppliers
    return ref_cs_sup;

function show_supplies
    return ref_cs_su;

function show_logs
    return ref_cs_log;
```

```
function show_products
    return ref_cs_pro is
    pro ref_cs_pro;
    begin
            open pro for
            select *
            from products
            order by pid asc;
            return pro;
    end;
```

2. Purchase saving function. Report the total saving of any purchase for any given pur#. The exception part can return the PL SQL error message to JDBC.

```
function purchase_saving(inputPur in purchases.pur#%type)
    return number
    is
    pur_saving number := 0;
    p_qty purchases.qty%type := 0;
    p_price products.original_price%type := 0;
    p_pid purchases.pid%type := null;
    p_total_price purchases.total_price%type := 0;

    begin
            select qty into p_qty
            from purchases
            where pur# = inputPur;

            select pid into p_pid
            from purchases
            where pur# = inputPur;

            select total_price into p_total_price
            from purchases
            where pur# = inputPur;

            select original_price into p_price
            from products
            where pid = p_pid;
            pur_saving := p_qty * p_price - p_total_price;
            return pur_saving;

    exception
            when no_data_found then
            raise_application_error(-20005, 'User-Defined Exceptions: PUR# is not found.');
            when others then
            raise_application_error(-20010, 'User-Defined Exceptions: PUR# format error. PUR# only contain numbers 0~999999.');


    end;
```

3. Monthly_sale_activities(employee_id) function. Report the sale information for any given employee. The exception part can return the user-defined error message to JDBC.

```
procedure monthly_sale_activities(
        employee_id in employees.eid%type,
        rst out ref_cs_monthSale)
as

begin
        open rst for
                select name, ceid, sqty, tprice, t
                from(

                (
                        select to_char(ptime,'MON/YYYY') as t, sum(qty) as sqty
                        from employees natural join purchases
                        where eid = employee_id
                        group by to_char(ptime, 'MON/YYYY')
                )
                natural join
                (
                        select to_char(ptime,'MON/YYYY') as t, sum(total_price) as tprice
                        from employees natural join purchases
                        where eid = employee_id
                        group by to_char(ptime, 'MON/YYYY')
                )
                natural join
                (
                        select to_char(ptime,'MON/YYYY') as t, count(eid) as ceid
                        from employees natural join purchases
                        where eid = employee_id
                        group by to_char(ptime, 'MON/YYYY')
                )),
                (select name from employees where eid = employee_id)
                order by t desc
                ;
eless in this procedure*/
exception
        when no_data_found then
        raise_application_error(-20006, 'User-Defined Exceptions: EID is not found.');
        when others then
        raise_application_error(-20009, 'User-Defined Exceptions: Only input right format EID(exx) can get the right search value');
end;
```

4. add_customer(c_id, c_name, c_telephone#) procedure uses for adding tuples to the customers table.

```
procedure add_customer(c_id in customers.cid%type,
        c_name in customers.name%type,
        c_telephone# in customers.telephone#%type)
as
begin
    insert into customers(cid,name,telephone#,last_visit_date) values (c_id,c_name,c_telephone#,sysdate);

exception
    when DUP_VAL_ON_INDEX then
        raise_application_error(-20007, 'User-Defined Exceptions: CID input duplication error. CID: cxxx');
    when others then
        raise_application_error(-20008, 'User-Defined Exceptions: CID/NAME/TELEPHONE# input format error. CID: cxxx/

end;
```

5. add_purchase(e_id, p_id, c_id, pur_qty) procedure uses for adding tuples to the purchases table. Adding tuple in purchases table is complicated. Because it evolved several other tables. Several events should be operated step by step in trigger part. In this procedure, I set one output variable (addValid) to notice JDBC if the purchase is valid. When the pur_qty larger than Qoh qty, JDBC should output related information, and this purchase is invalid.

```
        if (pur_qty > bufferQoh) then
            addValid := 0;
        else
            bufferTotalPrice := pur_qty * (1 - bufferDiscount) * bufferPrice;
            insert into purchases(eid,pid,cid,qty,ptime,total_price) values
               (e_id,p_id,c_id,pur_qty,sysdate,bufferTotalPrice);
            addValid := 1;

            select qoh into bufferQohNew
            from products
            where pid = p_id;
            if(bufferQoh - pur_qty != bufferQohNew) then
                addValid := 2;
            end if;
            new_qoh := bufferQohNew;
        end if;
```

6. Last procedure in this file is delete_purchase(pur#), using for deleting tuple in purchases table.

```
procedure delete_purchase(p_pur# in purchases.pur#%type)
as
buffer purchases.qty%type;
begin
    select qty into buffer
    from purchases
    where pur# = p_pur#;

    delete
    from purchases
    where pur# = p_pur#;

exception
    when no_data_found then
        raise_application_error(-20005, 'User-Defined Exceptions: PUR# is not found.');
    when others then
        raise_application_error(-20010, 'User-Defined Exceptions: PUR# format error. PUR# only contain numbers 0~999999.');

end;
```

C: Proj2data.sql

C_1:

This code contains three sequence _trigger generator (pur#, sup#, log#).

```
drop sequence seq_pur#;
drop sequence seq_sup#;
drop sequence seq_log#;


create sequence seq_pur#
increment by 1
start with 000001
maxvalue 999999
minvalue 000001
nocycle
order;

create sequence seq_sup#
increment by 1
start with 0001
maxvalue 9999
minvalue 0001
nocycle
order;

create sequence seq_log#
increment by 1
start with 00001
maxvalue 99999
minvalue 00001
nocycle
order;
```

```
create or replace trigger trg_pur#
----after insert on purchases
before insert on purchases
for each row
declare next_pur# number;
begin
    select seq_pur#.nextval into next_pur# from dual;
    :new.pur# := next_pur#;
end;
/

create or replace trigger trg_sup#
----after insert on supplies
before insert on supplies
for each row
declare next_sup# number;
begin
        select seq_sup#.nextval into next_sup# from dual;
        :new.sup# := next_sup#;
end;
/

create or replace trigger trg_log#
----after insert on logs
before insert on logs
for each row
declare next_log# number;
begin
        select seq_log#.nextval into next_log# from dual;
        :new.log# := next_log#;
end;
/
```

C_2:

1. When new customer is added in table, visits_made default value is 1.

```
create or replace trigger trg_customers_visits_made
----after insert on purchases
before insert on customers
for each row
begin
    :new.visits_made := 1;
end;
/
```

2. When new customer is added in table, a new tuple should be added in logs table.

```
create or replace trigger trg_logs_event1
after insert on customers
for each row
begin
    insert into logs(user_name,operation,op_time,table_name,tuple_pkey) values (u
end;
/
```

3. When customer's last_visit_date is updated, a new tuple should be added in logs table.

```
create or replace trigger trg_logs_event2
after update of last_visit_date on customers
for each row
begin
    insert into logs(user_name,operation,op_time,table_name,tuple_pkey) values (user,
end;
/
```

4. When new purchase tuple is added in purchases table, a new tuple should be added in logs table.

```
create or replace trigger trg_logs_event3
after insert on purchases
for each row
begin
    insert into logs(user_name,operation,op_time,table_name,tuple_pkey) values (user,
    --insert into supplies(pid,sid,sdate,quantity) values ('p001','s3',sysdate,234);
end;
/
```

5. When qoh in product is updated, a new tuple should be added in logs table.

```
create or replace trigger trg_logs_event4
after update of qoh on products
for each row
begin
    insert into logs(user_name,operation,op_time,table_name,tuple_pkey) values (
end;
/
```

6. When new supplies tuple is added in supplies table, a new tuple should be added in logs table.

```
create or replace trigger trg_logs_event5
after insert on supplies
for each row
begin
    insert into logs(user_name,operation,op_time,table_name,tuple_pkey) values (use
end;
/
```

7. When new tuple is added in purchases table, several events should be triggered.
   A. Update qoh in products table.
   B. Update visits_made in customers table.
   C. Update last_visit_date in customers table.
   D. If necessary, trigger a new supply (insert a new tuple in supplies table, when qoh is lower than qoh_threshold).
   E. If necessary, update qoh in products table again.

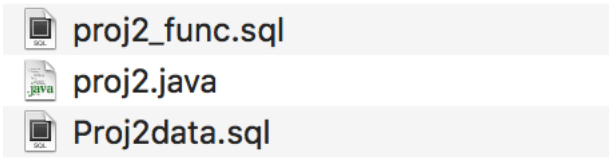
   All these things have clear relationship, so it is not a good idea to separate them in several triggers.

```
create or replace trigger trg_products_qoh
after insert on purchases
for each row
declare
        s_id char(2);
        p_id char(4);
        p_qoh number;
        p_qoh_h number;
begin
        p_id := :new.pid;
        update products
        set qoh = qoh - :new.qty
        where pid = :new.pid;

        update customers
        set visits_made = visits_made + 1
        where cid = :new.cid;

        update customers
        set last_visit_date = sysdate
        where cid = :new.cid;

        select qoh into p_qoh
        from products
        where pid = :new.pid;

        select qoh_threshold into p_qoh_h
        from products
        where pid = :new.pid;


        if(p_qoh < p_qoh_h) then
        select min(sid) into s_id
            from supplies
            where pid = p_id;
            insert into supplies(pid,sid,sdate,quantity) values (p_id,s_id,sys
            update products
            set qoh = qoh + (p_qoh_h + 10)
            where pid = :new.pid;
        end if;
end;
```

8. When delete a tuple in purchases table, a new tuple should be added in logs table. Moreover, customers table and products table should be update again, which will cause new insert in logs table.

```
create or replace trigger trg_delete_purchase
after delete on purchases
for each row
begin

        insert into logs(user_name,operation,op_time,table_name,tuple_pkey) value

        update customers
        set last_visit_date = sysdate
        where cid = :old.cid;

        update customers
        set visits_made = visits_made + 1
        where cid = :old.cid;

        update products
        set qoh = qoh + :old.qty
        where pid = :old.pid;
end;
```

D: How to Run demo:



1. Put all these three files in one path in Bingsuns server.
2. Compile java file:
   javac proj2.java
3. Enter sqlplus acad111 database.
   It must run Proj2data.sql first, then proj2_func.sql can be run.
   start Proj2data.sql
   start proj2_func.sql
4. Return to bingsuns server, run java code.
   java proj2
5. UI operation

   EXAMPLE #1: display employees table
   Enter java UI: (Enter your own ID and keyword)

```
bingsuns2% java proj2
Please enter user name >>> xzhan211
Please enter keyword >>>
```

```
**************************************************
|command       :                                   |
|1. [display]  : show the tuples in each table.    |
|2. [save]     : show purchase saving.             |
|3. [activity] : show monthly sale activity.       |
|4. [+customer]: add customer tuple.               |
|5. [+purchase]: add purchase tuple.               |
|6. [-purchase]: delete purchase tuple.            |
|7. [exit]     : exit JDBC                         |
|**************************************************|
|table name:                                       |
|1. employees                                      |
|2. customers                                      |
|3. products                                       |
|4. discounts                                      |
|5. suppliers                                      |
|6. supplies                                       |
|7. purchases                                      |
|8. logs                                           |
**************************************************

Please enter command: display
Please enter table name: employees
```

The top part is the hint of instructions, only 7 instructions.
The bottom part is the hint of table names.
There are two steps to operate this UI, enter the instruction first, then enter parameters such as table name or pur# etc.
Enter instruction (display) & parameter(employees)

Get the result: employees table

```
EID NAME            TELEPHONE#   EMAIL
-----------------------------------------------
e01 David           666-555-1234 david@rb.com
-----------------------------------------------
e02 Peter           777-555-2341 peter@rb.com
-----------------------------------------------
e03 Susan           888-555-3412 susan@rb.com
-----------------------------------------------
e04 Anne            666-555-4123 anne@rb.com
-----------------------------------------------
e05 Mike            444-555-4231 mike@rb.com
```

Other instruction information:
Follow the requirement document “proj2CS532-17”.
Instruction: display
Question 2 in requirement document.
As above operation.

Instruction: save
Question 3 in requirement document.

```
Please enter command: save
Please enter pur# number: 100002
The total saving for pur#100002 is $29.60
```

Instruction: activity
Question 4 in requirement document.

```
Please enter command: activity
Please enter employee_ID: e03

EID NAME            TIMES QUANTITY TOTAL_DOLLAR MONTH/YEAR
-------------------------------------------------------------
e03 Susan           1     2        35.91        SEP/2017
-------------------------------------------------------------
e03 Susan           3     5        493.13       OCT/2017
-------------------------------------------------------------
e03 Susan           1     1        17.96        AUG/2017
```

Instruction: +customer
Question 5 in requirement document.

```
Please enter command: +customer
Please enter customer cid: c020
Please enter customer name: Shaun
Please enter customer tel: 111-222-3333
```

Instruction: +purchase
Question 7 in requirement document.

```
Please enter command: +purchase
Please enter purchase eid: e03
Please enter purchase pid: p003
Please enter purchase cid: c002
Please enter purchase qty: 10
```

Instruction: -purchase
Question 8 in requirement document.

```
Please enter command: -purchase
Please enter delete purchase pur#: 100004
```