Database Application Development Assignment 2

Due: Monday, December 7 2020, 11:59pm

Objective:

In this assignment, you create a simple Retail application using the C++ programming language and Oracle (PL/SQL). This assignment helps students learn a basic understanding of application development using C++ programming and an Oracle database using PL/SQL.

Submission:

Your submission will be a single text-based .cpp file including your C++ program for the Database Application assignment.

`DBS311_ASS2_Lastname.cpp`

Your submission needs to be commented.

Instruction:

In this assignment, we use the same database that you have been using for the labs and the assignment 1.

Note: For each query in your assignment, make sure you handle the errors and display the proper message including the error code and the error message.

```
try{
        ...
}
catch (SQLException& sqlExcp) {
     cout << sqlExcp.getErrorCode() << ": " << sqlExcp.getMessage();
}
```

Declare the following structure before the ***main()*** function:

```
struct ShoppingCart {
    int product_id;
    double price;
    int quantity;
};
```

Connecting to an Oracle database from a C++ Program

In your function ***main()***, create a connection to your database.

First, declare the environment and the connection variables.

```
Environment* env = nullptr;
Connection* conn = nullptr;
```

Define and initialize the variable to store the username, password, and the host address.

```
string user = "username";
string pass = "password";
string constr = "myoracle12c.senecacollege.ca:1521/oracle12c";
```

Use the same Oracle username and password that you use for your labs and assignments. Create the environment and the connection. Make sure you handle any errors may be thrown as you program is executed.

```
env = Environment::createEnvironment(Environment::DEFAULT);
conn = env->createConnection(user, pass, constr);
```

Remember to terminate and close the connection and the environment, when your program terminates.

```
env->terminateConnection(conn);
Environment::terminateEnvironment(env);
```

After executing the statements make sure you terminate the statement.

```
conn->terminateStatement(stmt);
```

You will implement the following Oracle stored procedures and C++ functions:

Stored Procedures

find_customer (customer_id IN NUMBER, found OUT NUMBER);

This procedure has an input parameter to receive the customer ID and an output parameter named found.
This procedure looks for the given customer ID in the database. If the customer exists, it sets the variable *found* to 1. Otherwise, the *found* variable is set to 0.

To check if your query in the ***find_customer()*** procedure returns a row, you can check the no_data_found exception in the EXCEPTION block.

```
EXCEPTION
      WHEN no_data_found THEN
      found := 0;
```

find_product (product_id IN NUMBER, price OUT products.list_price%TYPE);

This procedure has an input parameter to receive the product ID and an output parameter named price.
This procedure looks for the given product ID in the database. If the product exists, it stores the product's list_price in the variable *price*. Otherwise, the *price* variable is set to 0.

```
EXCEPTION
      WHEN no_data_found THEN
      price := 0;
```

add_order (customer_id IN NUMBER, new_order_id OUT NUMBER)

This procedure has an input parameter to receive the customer ID and an output parameter named new_order_id.
To add a new order for the given customer ID, you need to generate the new order Id. To calculate the new order Id, find the maximum order ID in the orders table and increase it by 1.
This procedure inserts the following values in the orders table:
new_order_id
customer_id (input parameter)
'Shipped' (The value for the order status)
56 (The sales person ID)
sysdate (order date which is the current date)

add_order_item (orderId IN order_items.order_id%type,
itemId IN order_items.item_id%type,
productId IN order_items.product_id%type,
quantity IN order_items.quantity%type,
price IN order_items.unit_price%type)

This procedure has five IN parameters. It stores the values of these parameters to the table order_items.

C++ Functions

int mainMenu();

The ***mainMenu()*** function returns an integer value which is the selected option by the user from the menu. This function displays the following menu options:

1) Login
0) Exit

Prompt the user to choose an option. If the user enters the wrong value, ask the user to enter an option again until the user enters a valid options.

See the following example:

```
******************** Main Menu ********************
1)      Login
0)      Exit
Enter an option (0-1): 5
******************** Main Menu ********************
1)      Login
0)      Exit
You entered a wrong value. Enter an option (0-1):
```

If the user chooses option 1, ask the user to enter customer ID to login. To see if the customer with the entered ID exists, call the Oracle stored procedure ***find_customer()***. IF the value of the output parameter in the procedure is 1, let the customer to continue. If the value of the output parameter found is 0, call the ***mainMenu()*** functions again and asks the customer to login again. Continue this process until the user chooses the option 0 to exit or enters a valid customer ID.

int customerLogin(Connection conn, int customerId);*

Before you call this function, prompt the user to enter the customer ID.
Call this function in the ***main()*** function if the user chooses the login option from the main menu. This function receives an integer value as a customer ID and checks if the customer does exist in the database. This function returns 1 if the customer exists. If the customer does not exists, this function returns 0 and the main menu is displayed.
To validate the customer ID call the ***find_customer()*** stored procedure in this function.

See the following example:

```
******************** Main Menu ********************
1)      Login
0)      Exit
Enter an option (0-1): 1
Enter the customer ID: 1000
The customer does not exist.
******************** Main Menu ********************
1)      Login
0)      Exit
Enter an option (0-1): 1
Enter the customer ID: 44
-------------- Add Products to Cart --------------
Enter the product ID:
```

int addToCart(Connection conn, struct ShoppingCart cart[]);*

If the customerLogin() functions return 1 (The customer ID exists), call this function.
This function receives an OCCI pointer (a reference variable to an Oracle database) and an array of type ShoppingCart.

The customer can purchase up to five items in one order.
Write a loop to prompt the user to enter product IDs for the maximum of five products.

When the user enters the product ID in the *addToCart()* function, call the findProduct() functions to check if the product ID exists. IF the product exists, the function findProduct() returns the product's price. Display the product's price to the user and ask the user to enter the quantity.

If the user enters a valid product ID, display the following message and let the user to enter another product ID.

"Enter 1 to add more products or 0 to checkout: "

If the user chooses 1, ask the user to enter the next product ID. Otherwise, go to the next step to checkout. If the user enters 0, the function ***addToCart()***, returns the number of products (items) entered by the user.

For each product ID entered by the customer call the function ***findProduct()*** to see if the product ID exists.
If the ***findProduct()*** function returns 0 (The product ID does not exist), display a proper message and let the user enter the product ID again.

See the following example:

```
-------------- Add Products to Cart --------------
Enter the product ID: 1000
The product does not exists. Try again...
Enter the product ID: 900
The product does not exists. Try again...
Enter the product ID: 112
Product Price: 808.92
Enter the product Quantity: 3
Enter 1 to add more products or 0 to checkout: 1
Enter the product ID: 115
Product Price: 699.99
Enter the product Quantity: 2
Enter 1 to add more products or 0 to checkout: 0
```

double findProduct(Connection conn, int product_id);*

This function receives an OCCI pointer (a reference variable to an Oracle database) and an integer value as the product ID.

When the user enters the product ID in the ***addToCart()*** function, the function ***findProduct()*** is called.

This functions calls the ***find_product()*** Oracle stored procedure. The procedure receives the product ID and returns the price. If the price is 0, the product ID is not valid (does not exist). If the price is a non-zero value, it means the product ID is valid.

void displayProducts(struct ShoppingCart cart[], int productCount);

This function receives an array of type ShoppingCart and the number of ordered items (products). It display the product ID, price, and quantity for products stored in the cart array.

Call this function after the function ***AddToCart()*** to display the products added by the user to the shopping cart.

```
------- Ordered Products ---------
---Item 1
Product ID: 112
Price: 808.92
Quantity: 3
---Item 2
```

```
Product ID: 115
Price: 699.99
Quantity: 2
---------------------------------
Total: 3826.74
```

After displaying the products' information (product ID, price, and quantity), display the total order amount. To calculate the total order amount, first multiply the quantity and the price to calculate the total amount for each product. Next, sum up products' total amounts to calculate the total order amount.

*int checkout(Connection *conn, struct ShoppingCart cart[], int customerId, int productCount);*

Call this function after the function ***displayProduct()***.
This function receives an OCCI pointer (a reference variable to an Oracle database), an array of type ShoppingCart, an integer value as the customer ID, and an integer value as the number of ordered items (products).

First, display the following message:
`"Would you like to checkout? (Y/y or N/n) "`
If the user enters any values except "Y/y" and "N/n", display a proper message and ask the user to enter the value again.
`"Wrong input. Try again..."`

See the following example:

```
******************** Main Menu ********************
1)      Login
0)      Exit
Enter an option (0-1): 1
Enter the customer ID: 4
-------------- Add Products to Cart --------------
Enter the product ID: 112
Product Price: 808.92
Enter the product Quantity: 3
Enter 1 to add more products or 0 to checkout: 0
------- Ordered Products ---------
---Item 1
Product ID: 112
Price: 808.92
Quantity: 3
---------------------------------
Total: 2426.76
Would you like to checkout? (Y/y or N/n) t
```

```
Wrong input. Try again...
Would you like to checkout? (Y/y or N/n) 0
Wrong input. Try again...
Would you like to checkout? (Y/y or N/n) 1
Wrong input. Try again...
Would you like to checkout? (Y/y or N/n) y
The order is successfully completed.
******************** Main Menu ********************
1)      Login
0)      Exit
Enter an option (0-1): 0
Good bye!...
```

If the user enters "N/n", the function ***checkout()*** terminates and returns 0.

If the user enters "Y/y", the Oracle stored procedure ***add_order()*** is called. This procedure will add a row in the orders table with a new order ID (See the definition of the ***add_order()*** procedure.
This stored procedure returns an order ID, which will be used to store ordered items in the table order_items.

The item_id for the first product in the array is 1, for the second product is 2, and …
For all products in the array cart (*productCount* is the number of products stored in the array *cart*), call the stored procedure ***add_order_item()*** and pass the corresponding values to this stored procedure.

Sample execution:

```
******************** Main Menu ********************
1)      Login
0)      Exit
Enter an option (0-1): 5
******************** Main Menu ********************
1)      Login
0)      Exit
You entered a wrong value. Enter an option (0-1): 1
Enter the customer ID: 1000
The customer does not exist.
******************** Main Menu ********************
1)      Login
0)      Exit
Enter an option (0-1): 44
******************** Main Menu ********************
1)      Login
0)      Exit
You entered a wrong value. Enter an option (0-1): 1
```

```
Enter the customer ID: 44
-------------- Add Products to Cart --------------
Enter the product ID: 112
Product Price: 808.92
Enter the product Quantity: 2
Enter 1 to add more products or 0 to checkout: 1
Enter the product ID: 115
Product Price: 699.99
Enter the product Quantity: 3
Enter 1 to add more products or 0 to checkout: 0
------- Ordered Products ---------
---Item 1
Product ID: 112
Price: 808.92
Quantity: 2
---Item 2
Product ID: 115
Price: 699.99
Quantity: 3
---------------------------------
Total: 3717.81
Would you like to checkout? (Y/y or N/n) y
The order is successfully completed.
******************** Main Menu ********************
1)      Login
0)      Exit
Enter an option (0-1): 1
Enter the customer ID: 44
-------------- Add Products to Cart --------------
Enter the product ID: 110
Product Price: 3192.97
Enter the product Quantity: 2
Enter 1 to add more products or 0 to checkout: 1
Enter the product ID: 116
Product Price: 731.99
Enter the product Quantity: 1
Enter 1 to add more products or 0 to checkout: 1
Enter the product ID: 117
Product Price: 695.99
Enter the product Quantity: 3
Enter 1 to add more products or 0 to checkout: 0
------- Ordered Products ---------
---Item 1
Product ID: 110
Price: 3192.97
Quantity: 2
---Item 2
Product ID: 116
```

```
Price: 731.99
Quantity: 1
---Item 3
Product ID: 117
Price: 695.99
Quantity: 3
---------------------------------
Total: 9205.9
Would you like to checkout? (Y/y or N/n) n
The order is cancelled.
******************** Main Menu ********************
1)      Login
0)      Exit
Enter an option (0-1): 0
Good bye!...
```