**Database Application Development**

**Assignment 2 (30%)**

**Objective:**

Nesta atribuição, você cria um aplicativo de varejo simples usando a linguagem de programação C++ e Oracle (PL/SQL). Esta tarefa ajuda os alunos a aprender uma compreensão básica do desenvolvimento de aplicativos usando programação C++ e um banco de dados Oracle usando PL/SQL.

**Submission:**

***Sua submissão será baseada em texto .cpp incluindo seu programa C++ e um text-based .sql incluindo os procedimentos armazenados do Oracle para a atribuição do Aplicativo de Banco de Dados.***

DBS311sectioncode\_ASS2\_group#.cpp

DBS311sectioncode\_ASS2\_group#.sql

***Veja o exemplo a seguir:***

DBS311NAA\_ASS2\_group02.cpp

Seu envio precisa ser comentado.

Você envia:

O arquivo .sql inclui todos os procedimentos armazenados que você é solicitado a gravar na atribuição. Escreva procedimentos armazenados na mesma ordem em que são definidos no documento de atribuição. Teste todos os seus procedimentos e certifique-se de que eles podem ser chamados a partir do programa C++ sem erros.

O arquivo .cpp é o mesmo que application.cpp com duas funções adicionais que você define com base na instrução fornecida no documento de atribuição. O programa C++ enviado deve estar livre de erros e todas as funções devem funcionar com êxito. Teste seu programa antes de enviá-lo por e-mail.

**Esquema de marcação**

***A política de promoção do curso exige que todos os alunos concluam a tarefa com sucesso para passar no curso***.

Submissões com erros não serão consideradas como concluídas e receberão zero.

Se todos os procedimentos armazenados não forem criados com êxito e completamente, a atribuição não será considerada como concluída e não receberá a nota completa.

Se todos os procedimentos armazenados forem criados com êxito e o programa C++ for concluído e funcionar sem erros (Todas as funções são definidas e executam sua tarefa sem gerar erros), o envio da atribuição é considerado como concluído com êxito e recebe a nota completa.

|  |  |  |
| --- | --- | --- |
| Tasks | Weight | Total Weight |
| 8 Oracle stored procedures/Functions | 10% each | 80% |
| Two C++ functions | 10% each | 20% |
|  | **Total** | 100% |

Você pode ser solicitado a concluir e enviar a tarefa novamente depois que a tarefa for marcada. A marca que você recebe para a atribuição não será alterada para o novo envio. No entanto, talvez seja necessário concluir e enviar a tarefa para satisfazer os requisitos do curso.

**Instrução**

Nesta atribuição, usamos o mesmo banco de dados que você tem usado para os laboratórios e a atribuição 1.

**Nota**: Para cada consulta em sua atribuição, você manipula os erros e exibe a mensagem adequada, incluindo o código de erro e a mensagem de erro.

try{

...

}

catch (SQLException& sqlExcp) {

cout << sqlExcp.getErrorCode() << ": " << sqlExcp.getMessage();

}

A seguinte estrutura é declarada antes da  ***função*** main():

struct ShoppingCart {

int product\_id;

double price;

int quantity;

};

***Conectando-se a um banco de dados Oracle a partir de um programa C++***

Na função ***main(),*** uma conexão é criada com o banco de dados.

Primeiro, declaramos o ambiente e as variáveis de conexão.

Environment\* env = nullptr;

Connection\* conn = nullptr;

Também definimos e inicializamos a variável para armazenar o nome de usuário, a senha e o endereço do host.

string user = "username";

string pass = "password";

string constr = "myoracle12c.senecacollege.ca:1521/oracle12c";

Você usa o mesmo nome de usuário e senha Oracle que usa para seus laboratórios e atribuições.

Depois de criar o ambiente e a conexão, certificamo-nos de lidar com quaisquer erros que possam ser lançados à medida que o programa é executado.

env = Environment::createEnvironment(Environment::DEFAULT);

conn = env->createConnection(user, pass, constr);

Encerramos e fechamos a conexão e o ambiente, quando o programa termina.

env->terminateConnection(conn);

Environment::terminateEnvironment(env);

Depois de executar as instruções, certificamo-nos de que a declaração é encerrada.

conn->terminateStatement(stmt);

Você implementará os seguintes procedimentos armazenados Oracle e os chamará nas seguintes funções C++::

***Procedimentos armazenados***

*find\_customer (customer\_id EM NÚMERO, DESCOBERTO NÚMERO);*

Este procedimento tem um parâmetro de entrada para receber o ID do cliente e um parâmetro de saída chamado found.

Este procedimento procura o ID do cliente fornecido no banco de dados. Se o cliente existir, ele define a variável *encontrada* como 1. Caso contrário, a variável *encontrada* será definida como 0.

Para verificar se a sua consulta no ***find\_customer()***procedimento retorna uma linha, você precisa verificar a exceção no\_data\_found no bloco EXCEPTION.

EXCEÇÃO

WHEN no\_data\_found THEN

found := 0;

Para verificar se a sua consulta no ***find\_customer()*** procedimento retorna várias linhas, você precisa verificar o too\_many\_rows no bloco EXCEPTION e exibir uma mensagem adequada.

Para detectar outros erros, verifique a exceção OTHERS no bloco EXCEPTION e exiba uma mensagem apropriada.

*find\_product (productId IN NUMBER,*

*price OUT products.list\_price%TYPE,*

*productName OUT products.product\_name%TYPE);*

Este procedimento tem um parâmetro de entrada para receber a ID do produto e um parâmetro de saída chamado price.

Este procedimento procura a ID do produto fornecida no banco de dados. Se o produto existir, ele armazena o product\_name do produto na variável *productName* e o list\_price do produto na variável *price*. Se o produto não existir, o productName será definido como null e a variável *price* será definida como 0.

Em novembro e dezembro, a empresa dá 10% de desconto em todos os produtos das categorias 2 e 5. Antes de armazenar o preço de tabela dos produtos que pertencem às categorias 2 e 5 no preço variável, calcule o novo preço de tabela se o mês atual for novembro ou dezembro e envie de volta o novo preço de tabela para o chamador como preço.

EXCEPTION

WHEN no\_data\_found THEN

price := 0;

Para verificar se sua consulta no  ***procedimento find\_product()*** retorna várias linhas, você precisa verificar a exceção too\_many\_rows no bloco EXCEPTION e exibir uma mensagem adequada.

Para detectar outros erros, verifique a exceção OTHERS no bloco EXCEPTION. e exibir uma mensagem adequada.

*add\_order (customer\_id IN NUMBER, new\_order\_id OUT NUMBER)*

Este procedimento tem um parâmetro de entrada para receber o ID do cliente e um parâmetro de saída chamado new\_order\_id.

Para adicionar um novo pedido para o ID do cliente fornecido, você precisa gerar o novo ID do pedido.

Para calcular a nova ID da ordem, chame a função generate\_order\_id(). O procedimento add\_order insere os seguintes valores na tabela de pedidos.

new\_order\_id (as 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)

*generate\_order\_id ()*

***Esta é uma função Oracle sem parâmetros.*** Ele localiza o ID máximo do pedido na tabela de pedidos e o aumenta em 1 como novo ID do pedido. A função ***retorna*** o novo ID do pedido para o chamador.

*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)*

Este procedimento tem cinco parâmetros IN. Ele armazena os valores desses parâmetros na tabela order\_items.

*customer\_order (customerId IN NUMBER, orderId IN OUT NUMBER)*

Este procedimento recebe dois valores como ID do cliente e ID do pedido e confirma se existe algum pedido com esse ID do pedido para esse cliente na tabela de pedidos. Se o ID do pedido com esse ID do cliente existir, o procedimento passará o ID do pedido para o chamador. Caso contrário, ele passa 0 para o chamador.

*display\_order\_status(orderId IN NUMBER, status OUT orders.status%type)*

Este procedimento tem um parâmetro de entrada para receber um ID de pedido e um parâmetro de saída para passar o status do pedido para o chamador. SE o ID do pedido de recebimento existir, o procedimento armazenará o status do pedido na  *variável de status*. Se o ID do pedido não existir, armazene null na  *variável* de status.

*cancel\_order (orderId IN NUMBER, cancelStatus OUT NUMBER)*

Este procedimento tem um parâmetro de entrada para receber um ID de pedido e um parâmetro de saída para passar um valor para o chamador. SE o ID do pedido de recebimento existir, o procedimento armazenará o valor do status da coluna em uma variável *orderStatus. Se orderStatus for 'Canceled', o procedimento armazenará 1 no parâmetro cancelStatus*. Se o *orderStatus* for enviado, o procedimento armazenará 2 no parâmetro *cancelStatus*. Caso contrário, ele armazena 3 no parâmetro *cancelStatus* e atualiza o status desse pedido para "Cancelado". Se o ID do pedido não existir, ele armazenará 0 na variável cancel.

O parâmetro *cancelStatus* obtém os seguintes valores:

0: The order does not exit.

1: The order has been already canceled.

2: The order is shipped and cannot be canceled.

3: The order is canceled successfully.

O procedimento armazenado não imprime nenhuma saída.

***C++ Functions***

Leia a definição de funções a seguir para saber como o código C++ funciona. Você precisa chamar os procedimentos armazenados que você escreveu na seção anterior nessas funções com base na instrução a seguir. Se o procedimento armazenado passar valores para o programa, defina variáveis ou use variáveis definidas no programa específico para armazenar valores de passagem do procedimento armazenado.

**Sua tarefa nesta seção é adicionar duas funções:**

*void displayOrderStatus(Connection\* conn, int orderId, int customerId);*

*void cancelOrder(Connection\* conn, int orderId, int customerId);*

***void displayOrderStatus(Connection\* conn, int orderId, int customerId);***

Essa função chama o procedimento armazenado customer\_order para confirmar que o ID do pedido inserido pertence ao cliente. Se o valor do segundo parâmetro do procedimento customer\_order for um valor diferente de zero, o ID do pedido pertencerá ao cliente.

Se o valor do segundo parâmetro do procedimento customer\_order for zero, exiba a seguinte mensagem:

“Order ID is not valid.”

Se o ID do pedido for válido, chame o *display\_order\_status* para receber o status do pedido armazenado no segundo parâmetro deste procedimento armazenado. Se o status do pedido for nulo, exiba a seguinte mensagem:

“Order does not exist.”

Se o status não for nulo, exiba uma mensagem apropriada.

Veja a mensagem de exemplo:

“Order is shipped.”

***void cancelOrder(Connection\* conn, int orderId, int customerId);***

Esta função chama o *cancel\_order* procedimento armazenado para confirmar que o ID do pedido inserido pertence ao cliente. Se o valor do segundo parâmetro do procedimento customer\_order for um valor diferente de zero, o ID do pedido pertencerá ao cliente.

Se o valor do segundo parâmetro do procedimento *customer\_order* for zero, exiba a seguinte mensagem:

“Order ID is not valid.”

0: The order does not exit.

1: The order has been already canceled.

2: The order is shipped and cannot be canceled.

3: The order is canceled successfully.

Se o ID do pedido for válido, chame o *cancel\_order* procedimento para o pedido com o ID do pedido passado para esta função. Se o valor passado pelo segundo parâmetro do *cancel\_order* for 1, exiba:

“The order has been already canceled.”

Se o valor passado pelo segundo parâmetro do *cancel\_order* for 2, exiba:

“The order is shipped and cannot be canceled.”

Se o valor passado pelo segundo parâmetro do *cancel\_order* for 3, exiba:

“The order is canceled successfully.”

As seguintes funções existem no arquivo application.cpp. Você modificará esse arquivo para adicionar duas funções C++ descritas acima.

*void displayOrderStatus(Connection\* conn, int orderId, int customerId);*

*cancelOrder(Connection\* conn, int orderId, int customerId);*

*int mainMenu();*

A função ***mainMenu()*** retorna um valor inteiro que é a opção selecionada pelo usuário no menu. Esta função exibe as seguintes opções de menu:

1. Login
2. Exit

O programa solicita que o usuário escolha uma opção. Se o usuário inserir o valor errado, o programa pedirá que o usuário insira uma opção novamente até que o usuário insira uma opção válida.

Veja o exemplo a seguir:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 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):

Se o usuário escolher a opção 1, o programa solicitará que o usuário insira o ID do cliente para fazer login. Para ver se o cliente com o ID inserido existe, o programa chama o procedimento armazenado Oracle ***find\_customer()***. SE o valor do parâmetro de saída no procedimento for 1, o programa permitirá que o cliente continue. Se o valor do parâmetro de saída encontrado for 0, o programa chama a função ***mainMenu()*** novamente e pede ao cliente para fazer login novamente. O programa continua esse processo até que o usuário escolha a opção 0 para sair ou insira um ID de cliente válido.

*int subMenu();*

A função ***subMenu()*** retorna um valor inteiro que é a opção selecionada pelo usuário no menu. Esta função exibe as seguintes opções de menu:

1) Place an order

2) Check an order status

3) Cancel an order

0) Exit

Se o usuário escolher a opção 1, ele poderá fazer um pedido. Se o usuário escolher a opção 2, a função displayOrderStatus será chamada. Se o usuário escolher a opção 3, o programa chamará a função cancelOrder. Se o usuário escolher a opção 0, o usuário retornará ao menu principal.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Customer Service Menu \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1) Place an order

2) Check an order status

3) Cancel an order

0) Exit

Enter an option (0-3):

*int customerLogin(Connection\* conn, int customerId);*

Antes de chamar essa função, o programa solicita que o usuário insira o ID do cliente.

Esta função é chamada na função ***main()*** se o usuário escolher a opção de login no menu principal. Essa função recebe um valor inteiro como uma ID de cliente e verifica se o cliente existe no banco de dados. Essa função retorna 1 se o cliente existir. Se o cliente não existir, essa função retornará 0 e o menu principal será exibido.

Para validar a ID do cliente, chame o  ***procedimento/função armazenado*** find\_customer() nesta função.

Veja o exemplo a seguir:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 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[]);*

Se as funções customerLogin() retornarem 1 (O ID do cliente existe), essa função será chamada.

Essa função recebe um ponteiro OCCI (uma variável de referência para um banco de dados Oracle) e uma matriz do tipo ShoppingCart.

O cliente pode comprar até cinco itens em um único pedido.

Em um loop, o programa solicita que o usuário insira IDs de produto para o máximo de cinco produtos.

Quando o usuário insere a ID do produto na função *addToCart(),* a função findProduct() é chamada para verificar se a ID do produto existe. SE o produto existir, a função findProduct() retornará o preço do produto. O programa exibe o preço do produto para o usuário e pede que o usuário insira a quantidade.

Se o usuário inserir uma ID de produto válida, o programa exibirá a seguinte mensagem e permitirá que o usuário insira outra ID de produto.

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

Se o usuário escolher 1, o programa pedirá que o usuário insira a próxima ID do produto. Se o usuário digitar 0, a função ***addToCart()***, retornará o número de products (items) inserido pelo usuário.

Para cada ID de produto inserido pelo cliente, a função ***findProduct()*** é chamada para ver se o product ID existe.

Se a função ***findProduct()*** retornar 0 (A ID do produto não existe), o programa exibirá uma mensagem apropriada e permitirá que o usuário insira a ID do produto novamente.

Veja o exemplo a seguir:

-------------- 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);*

Essa função recebe um ponteiro OCCI (uma variável de referência para um banco de dados Oracle) e um valor inteiro como a ID do produto.

Quando o usuário insere a ID do produto na função ***addToCart(),*** a função ***findProduct()*** é chamada.

Essa função chama o procedimento/função armazenada ***find\_product()*** Oracle. O procedimento recebe a ID do produto e devolve o preço. Se o preço for 0, a ID do produto não é válida (não existe). Se o preço for um valor diferente de zero, significa que a ID do produto é válida.

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

Esta função recebe uma matriz do tipo ShoppingCart e o número de itens solicitados (produtos). Ele exibe o ID do produto, o preço e a quantidade dos produtos armazenados no conjunto do carrinho.

Esta função é chamada após a função ***AddToCart()*** para exibir os produtos adicionados pelo usuário ao carrinho de compras.

------- 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

Depois de exibir as informações dos produtos (ID do produto, preço e quantidade), o programa exibe o valor total do pedido. Para calcular o valor total do pedido, o programa primeiro multiplica a quantidade e o preço para calcular o valor total de cada produto. Em seguida, ele soma os valores totais dos produtos para calcular o valor total do pedido.

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

Essa função é chamada após a função ***displayProduct()***.

Essa função recebe um ponteiro OCCI (uma variável de referência para um banco de dados Oracle), uma matriz do tipo ShoppingCart, um valor inteiro como o ID do cliente e um valor inteiro como o número de itens solicitados (products).

Primeiro, o programa exibe a seguinte mensagem:

"Would you like to checkout? (Y/y or N/n) "

Se o usuário inserir quaisquer valores, exceto "Y/y" e "N/n", o programa exibirá uma mensagem apropriada e pedirá que o usuário insira o valor novamente.

"Wrong input. Try again..."

Veja o exemplo a seguir:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 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...!

Se o usuário digitar "N/n", a função ***checkout()*** será encerrada e retornará 0.

Se o usuário digitar "S/y", o procedimento armazenado Oracle ***add\_order()*** será chamado. Esse procedimento adicionará uma linha na tabela de pedidos com uma nova ID de pedido (consulte a definição do  ***procedimento*** add\_order().

Esse procedimento armazenado retorna uma ID do pedido, que será usada para armazenar itens solicitados na tabela order\_items.

O item\_id para o primeiro produto na matriz é 1, para o segundo produto é 2, e ...

Para todos os produtos no carrinho de matriz (*productCount* é o número de produtos armazenados no carrinho de matriz), o programa chama o procedimento armazenado ***add\_order\_item()*** e passa os valores correspondentes para esse procedimento armazenado.

Exemplo de execução:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 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...!