CSIS05I

Database Systems II

Lab (7)

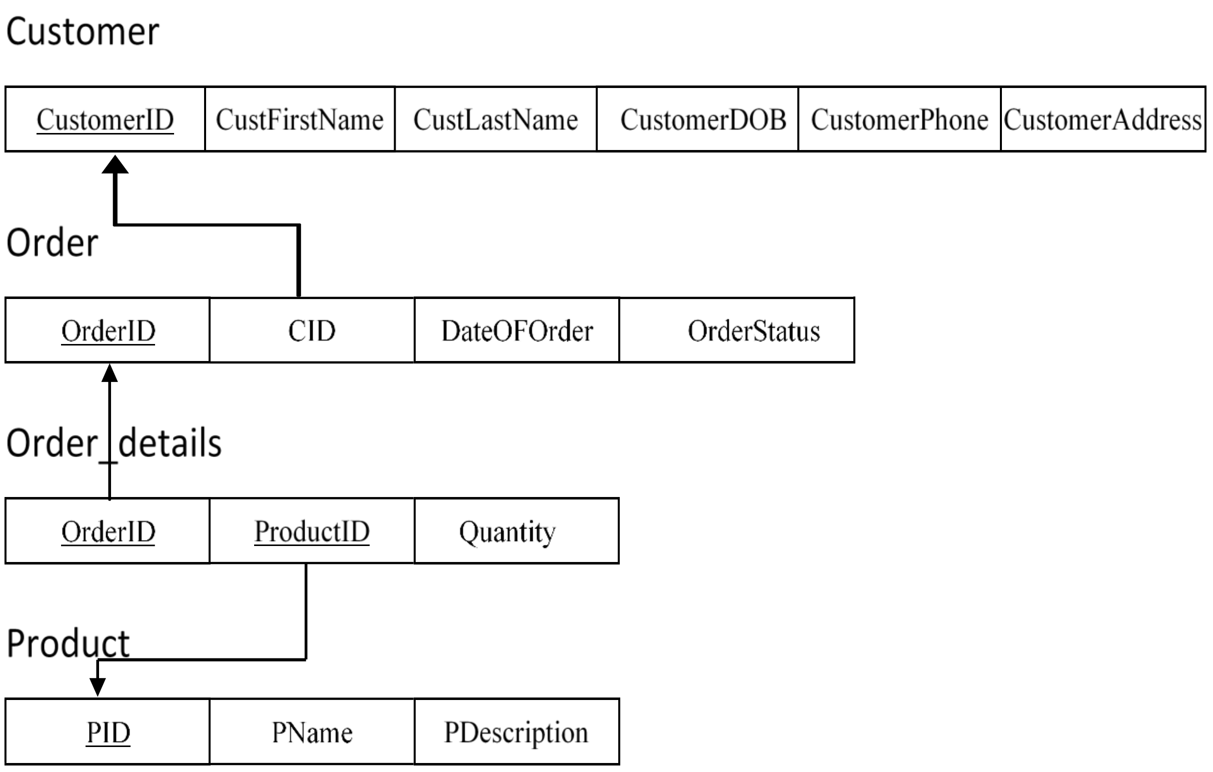
INTRODUCTION TO STORED PROCEDURES

Lab (7)

The Overview

Through this lab, we will introduce the stored procedure in SQL. We will learn how to create it and execute it. We will use again the store schema database.

Store Schema:



Stored Procedure:

A **Stored Procedure** is a way for you to store a set of SQL statements and accompanying programming statements within the database and run them later. Stored procedures come in handy as they allow you combine both procedural logic as well as SQL statements.

A **Stored Procedure** is a type of code in SQL that can be stored for later use and can be used many times. So, whenever you need to execute the query, instead of calling it you can just call the stored procedure. You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter values that are passed. In other words, it is nothing but the group of SQL statements that accepts some input in the form of parameters and performs some task.

Create procedure:

To define our procedure, we use the CREATE PROCEURE command. The general format for the command is CREATE PROCEDURE procedure-name.

```
CREATE PROCEDURE Procedure_name
```

Once you have declared the procedure name, you can declare and input parameters:

- ✓**Input parameters:** are used to provide the procedure with values.

```
CREATE PROCEDURE Procedure_name
                @parameter1 datatype,
                @parameter2 datatype,
                ……….
AS
        sql_statement / code
GO
```

To execute the stored procedure:

```
EXEC procedure_name;
```

OR

```
Execute procedure_name;
```

Note: If the stored procedure has been executed, you can't modify in it and run it again, to do that you have to alter the procedure.

Modifying the procedure

To modify the procedure after execution, you need to use the key word ALTER to do any changes in it

```
ALTER PROCEDURE procedure_name
            @parameter1 datatype,
            @parameter2 datatype,
            .........
AS
      sql_statement / code
GO
```

Note: *You can also use:* Create or alter procedure *procedure-name….*

Example 1:

Retrieve all the information from customer using stored procedure.

```
CREATE PROCEDURE SelectAllCustomers
AS
SELECT * FROM Customer
GO

Execute SelectAllCustomers;
```

Example 2:

Retrieve all information about the customer whose last name is entered by the user and named John using stored procedure.

```
CREATE PROCEDURE SelectAllLastNamenew
@CustLName varchar(20)
AS

SELECT *
FROM Customer
where CustLastName = @CustLName
GO

Execute SelectAllLastNamenew @CustLName = 'John';
```

Example 3:

Retrieve the customer full name, order ID and order status for the customers who have their orders delivered. Null values can be returned for the orders.

```
CREATE PROCEDURE SelectcustomerDetails

AS
SELECT custfirstname + ' ' + custlastname as 'Customer Name',
OrderID, OrderStatus
FROM Customer left outer join orderr
on CustomerID = CID
where OrderStatus = 'Delivered';


Execute SelectcustomerDetails;
```

Example 4:

Retrieve the customer whose name is Mark and his Id is 1 and lives in New Cairo.

```
CREATE PROCEDURE Selectcustomer1

AS

SELECT *
FROM Customer
where CustomerID = '1' and CustFirstName='Mark' and CustomerAddress='New
Cairo';

GO

Execute Selectcustomer1;
```

<u>OR</u> we can make the user enter the data himself by using the parameters

```
CREATE PROCEDURE Selectcustomer
@CustID int,
@CustFN varchar(30),
@CustAddress varchar(30)

AS

SELECT *
FROM Customer WHERE CustomerID = @CustID AND CustFirstName = @CustFN AND
CustomerAddress = @CustAddress

GO
```

You can execute the procedure using one of the following lines. Both of them will retrieve the same result.

```
Execute Selectcustomer @CustID = '1', @CustFN= 'Mark', @CustAddress= 'New
Cairo';

OR

Execute Selectcustomer '1', 'Mark', 'New Cairo';
```

Example 5:

Retrieve all the order IDs along with the product ID and Product description where the product description includes 'cam' and product quantity is more than 3.

```
CREATE PROCEDURE OrderdetailsInofrmation

AS

SELECT OrderID, PID, PDescription
FROM Order_details join Product
on ProductID=PID
where Pdescription like '%cam%' and Quantity > 3;

GO

Execute OrderdetailsInofrmation;
```

Example 6:

Retrieve the first name and address of all customers except the customers that have made orders on 26/7/2019, using stored procedure and nested queries.

```
CREATE PROCEDURE OrderDate

AS

SELECT CustFirstName, CustomerAddress
FROM Customer
WHERE  NOT EXISTS ( SELECT * FROM Orderr WHERE CID = CustomerID AND
DateOFOrder = '26/7/2019' );

GO

Execute OrderDate;
```

OR

```
CREATE PROCEDURE OrderDate1

AS

SELECT CustFirstName, CustomerAddress
FROM Customer
WHERE CustomerID  NOT IN ( SELECT CID FROM Orderr WHERE DateOFOrder =
'26/7/2019' );

GO

Execute OrderDate1;
```

Example 7:

Retrieve the product name where the order status is 'Not Delivered', using stored procedure and nested queries.

```
CREATE PROCEDURE Pname_Order

AS

Select Pname
from Product
where PID in (Select ProductID
              from order_details join Orderr
              on order_details.OrderID = orderr.OrderID
              where orderr.OrderStatus = 'NotDelivered');
GO

Execute Pname_Order;
```

Example 8:

Update the previous query to retrieve both the product name and product id, whose status is delivered or not delivered based on the user input.

```
ALTER PROCEDURE Pname_Order
@status varchar(50)

AS

Select Pname, PID from Product
where PID in (Select ProductID
from order_details join Orderr
on order_details.OrderID = orderr.OrderID
where orderr.OrderStatus = @status);

GO

Execute Pname_Order 'delivered';

Execute Pname_Order 'notdelivered';
```

Example 9:

Retrieve customer IDs along with the total number of their orders; only retrieve the customers who have made more than one order. Note, please add this insert statement before you test example 9.

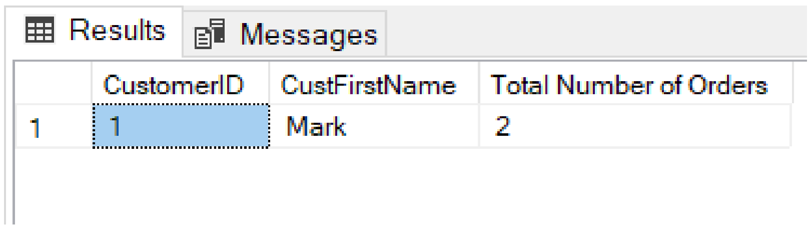
```
INSERT INTO Orderr VALUES('104', '1', '26/7/2019', 'Delivered');

CREATE PROCEDURE CountingOrders
@count int = 1

AS

Select CustomerID, CustFirstName, COUNT (OrderID) AS 'Total Number of Orders'
from Customer , Orderr
where CustomerID = CID
Group by CustomerID, CustFirstName
Having COUNT (OrderID) > @count;

Execute CountingOrders;
```



	CustomerID	CustFirstName	Total Number of Orders
1	1	Mark	2

Exercise

1- Retrieve the customer ids and names in which their DOB is ‘18/12/1960’ along with its corresponding order id using a stored procedure.

2- Retrieve product ID and description in which the quantity of the product should be greater than five using a stored procedure.

3- Retrieve the customer ID and the customer’s order in which the customer last name should end by ‘n’ using a stored procedure.

4- Retrieve the order ID and its status along with the product name using a stored procedure.

5- Retrieve the product names and the order ID that has been ordered by customer his first name is ‘Steven’ (entered by the user) using a stored procedure and nested queries.