CSIS05I

Database Systems II

Lab (8)

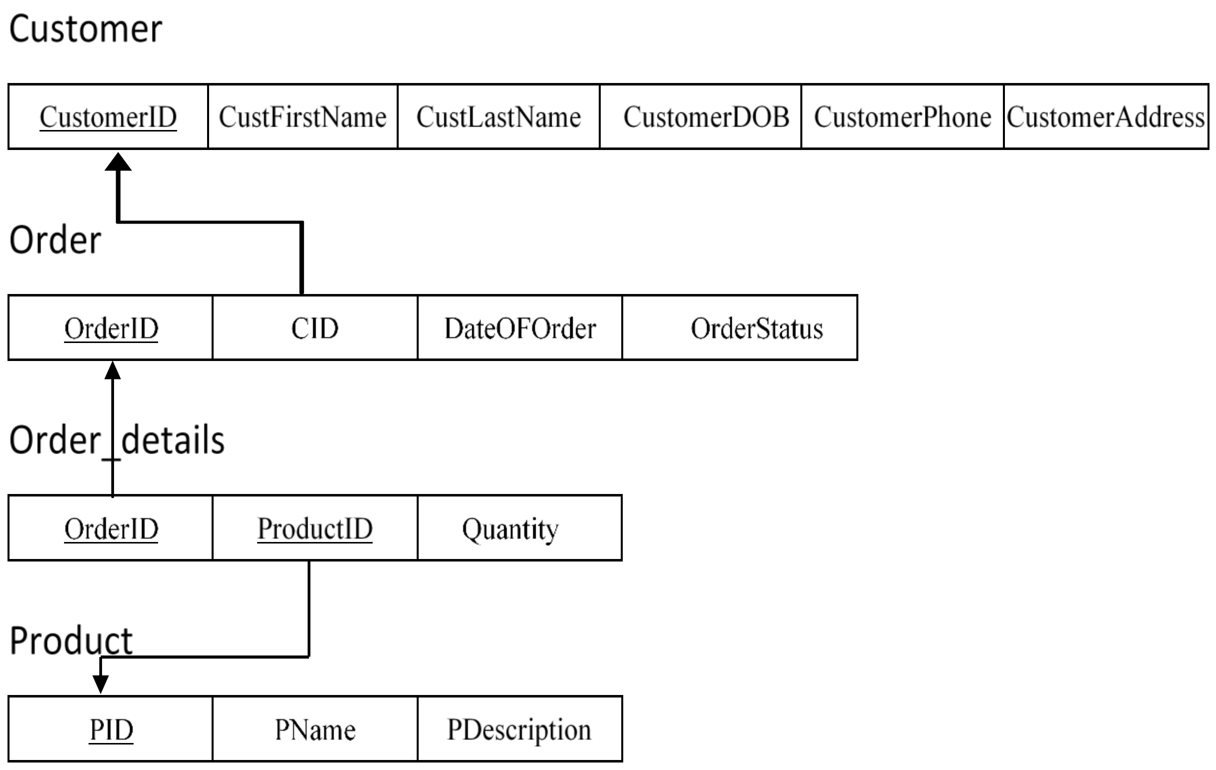
INTRODUCTION TO STORED PROCEDURES II

Lab (8)

The Overview

In the previous lab, we mentioned the simple stored procedure and the creation syntax for it. In addition, we mentioned how to modify in the procedure after execution using alter. Through this lab, we will introduce the stored procedure in SQL in a complex way and its parameter types.

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. This makes SQL Server very flexible, as SQL by itself is not suitable to tackle all problems, and being able to call upon procedural logic to string together multiple SQL statements into one step is handy.

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

To drop a procedure, we use DROP PROCEDURE command. The general format is:

DROP PROCEDURE Procedure_name

Once you have declared the procedure name, you can declare parameters. There are many types of the parameters as:

- ✓ **Input parameters:** used to provide the procedure with values given in the SQL statement. There are two types of it:
  - **User defined parameter**
  - **Default parameter**
- ✓ **Output parameter:** used to print out a value to the user. A Stored Procedure can have any number of output parameters.

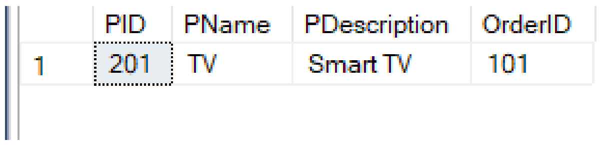
User defined parameter:

The user-defined parameter created in the SQL Server stored procedure that accepts the input parameters and processes the records based on the input parameter (entered by the user).

Example 1:

Retrieve the product ID, description, name and the order ID given that the user can pass different values in the product ID anytime.

```
CREATE PROCEDURE GetProductDesc_withparameters
@ProductID INT
AS
SELECT PID, PName, PDescription, OrderID
FROM Product, Order_Details
WHERE PID = ProductID
and PID=@ProductID
Go

EXEC GetProductDesc_withparameters 201;
```



	PID	PName	PDescription	OrderID
1	201	TV	Smart TV	101

Default parameter:

The default parameter is stated from the beginning inside the stored procedure SQL code. The SQL will have stored procedure with default parameter values.

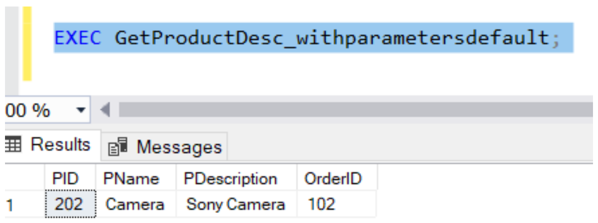
Example 2:

Retrieve the product ID, description, name and the order ID given that the product ID must be 202.

```
CREATE PROCEDURE GetProductDesc_withparametersdefault
@ProductID INT =202
AS
SELECT PID, PName, PDescription, OrderID
FROM Product, Order_Details
WHERE PID = ProductID
and PID=@ProductID
go


EXEC GetProductDesc_withparametersdefault;
```

<u>Note:</u> When we execute the above procedure without passing the parameter value, the default value will be used. However, if you passed another value while execution, the default value will be ignored and the passed value will be considered as the new parameter value.

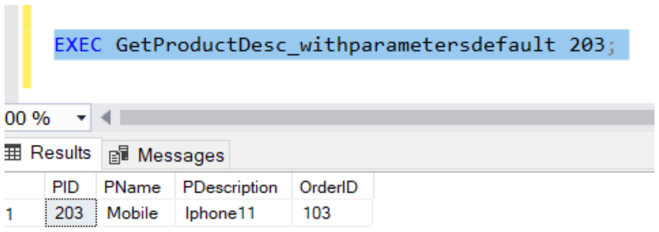
The default value which is 202 in the procedure:



```
EXEC GetProductDesc_withparametersdefault;
```

	PID	PName	PDescription	OrderID
1	202	Camera	Sony Camera	102

If you passed another value:



```
EXEC GetProductDesc_withparametersdefault 203;
```

	PID	PName	PDescription	OrderID
1	203	Mobile	Iphone11	103

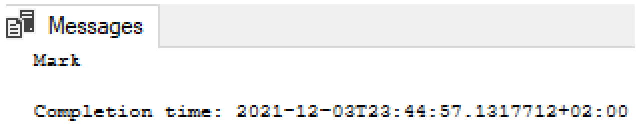
Output parameter:

The output parameter used if you want to show a value to the user. As with the help of the output parameters, the stored procedure can pass a data value back to the user.

Example 3:

Retrieve the first name of the customer whose ID is 1 (note that user must pass the ID value).

```
Create PROCEDURE Usp_GetCustomerDetails
--Input parameters
@CustId int,
--Output parameters
@FirstName varchar(100) OUT
AS

select @FirstName = (select CustFirstName
FROM Customer
where customerID =@CustId)

Print @FirstName

GO
Execute Usp_GetCustomerDetails 1 , OUT
```

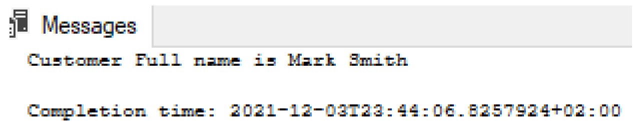


```
Messages
Mark

Completion time: 2021-12-03T23:44:57.1317712+02:00
```

Example 4:

Update the previous procedure and retrieve the full name of the customer whose ID is 1 (note that user must pass the ID value).

```
Alter PROCEDURE Usp_GetCustomerDetails
--Input parameters
@CustId int,
--Output parameters
@FullName varchar(100) OUT
AS
select @FullName = (select CustFirstName+' '+CustLastName
FROM Customer
where customerID =@CustId)
Print 'Customer Full name is ' + @FullName
GO
Execute Usp_GetCustomerDetails 1 , OUT
```



```
Messages
Customer Full name is Mark Smith

Completion time: 2021-12-03T23:44:06.8257924+02:00
```

***Example** 5*:

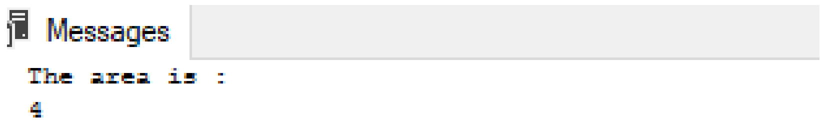
The following example shows you a simple calculation. The following example is general and not related to the store database schema, as it shows you how to calculate something and let the parameter carry this calculation. The question wants to get the value of the area, So, we have to create an output parameter that will carry the calculation and return it back to the user.

Calculate the area of the rectangle given that the user will define any two numbers to get the area.

```
CREATE PROCEDURE CalcArea
@height float,
@width float,
@area float OUT
AS
SELECT @area=@height*@width;
Go
```

To Call the procedure:

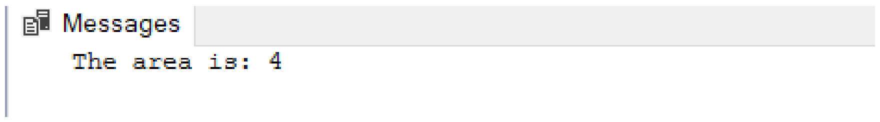
```
DECLARE @Area1 float
Execute CalcArea 2, 2 , @Area1 OUT
Print 'The area is :'
Print @Area1
```



```
The area is :
4
```

Another way of calling the procedure:

```
DECLARE @Area1 float
Execute CalcArea 2, 2 , @Area1 out
Print 'The area is: ' + CAST(@Area1 as varchar)
```



```
The area is: 4
```

Insert stored procedure

A stored procedure can be used to allow the user to insert values to tables using input parameters.

Example 6:

Insert new records to table Customer using stored procedure input parameters by the user. (You can specify different values in each run).

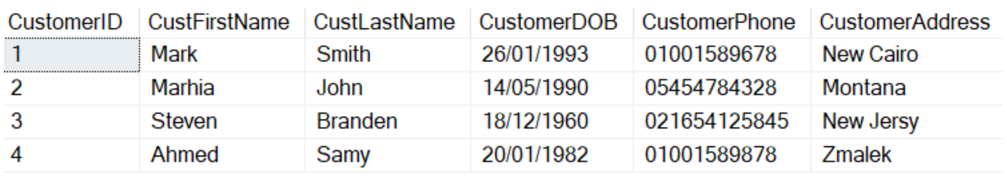
```
Create PROCEDURE CustomerInsertion
@CustomerID int,
@FirstName varchar(30),
@LastName varchar(30),
@DOB varchar(30),
@phone varchar(30),
@CustAddress varchar(30)
AS
insert into Customer
(CustomerID, CustFirstName, CustLastName, CustomerDOB, CustomerPhone,
CustomerAddress)
values
(@CustomerID, @FirstName, @LastName, @DOB, @phone, @CustAddress)
go
```

To Call the procedure:

```
Exec CustomerInsertion '4', 'Ahmed', 'Samy', '20/01/1982', '01001589878', 'Zmalek'
```

To test it,

```
select* from Customer;
```



CustomerID	CustFirstName	CustLastName	CustomerDOB	CustomerPhone	CustomerAddress
1	Mark	Smith	26/01/1993	01001589678	New Cairo
2	Marhia	John	14/05/1990	05454784328	Montana
3	Steven	Branden	18/12/1960	021654125845	New Jersy
4	Ahmed	Samy	20/01/1982	01001589878	Zmalek

Example 7:

Insert the records of table Customer and table order into one new table using stored procedure.

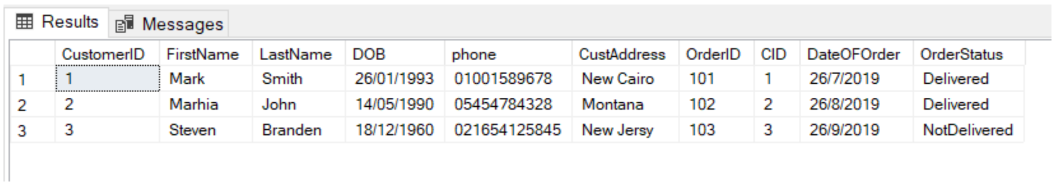
```
Create table CustomerOrderTable(
CustomerID int,
FirstName varchar(30),
LastName varchar(30),
DOB varchar(30),
phone varchar(30),
CustAddress varchar(30),
OrderID INT,
CID INT,
DateOFOrder VARCHAR(30),
OrderStatus varchar(30)
);


CREATE PROCEDURE CustomerAndOrderInsertion
AS
insert into CustomerOrderTable
select *
from Customer, Orderr
where CID = CustomerID
go
```

To Call the procedure:

```
exec CustomerAndOrderInsertion
```

To test it,

```
select* from CustomerOrderTable;
```



Results Messages

	CustomerID	FirstName	LastName	DOB	phone	CustAddress	OrderID	CID	DateOFOrder	OrderStatus
1	1	Mark	Smith	26/01/1993	01001589678	New Cairo	101	1	26/7/2019	Delivered
2	2	Marhia	John	14/05/1990	05454784328	Montana	102	2	26/8/2019	Delivered
3	3	Steven	Branden	18/12/1960	021654125845	New Jersy	103	3	26/9/2019	NotDelivered

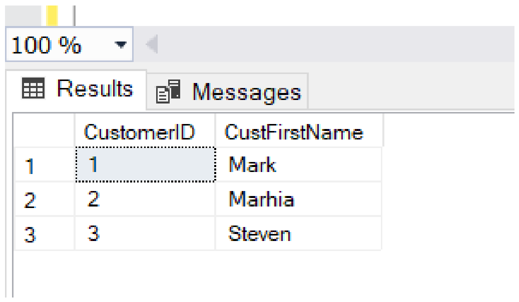
Encrypted stored procedure

We can hide the source code in the stored procedure by creating the procedure with the “ENCRYPTION” option.

Example 8:

Hide all the information related to the customer ID and First name.

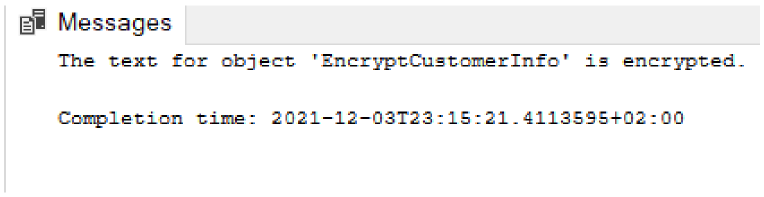
```
CREATE PROCEDURE EncryptCustomerInfo
WITH ENCRYPTION
AS
SELECT CustomerID,CustFirstName
FROM Customer
Go

exec EncryptCustomerInfo
```



	CustomerID	CustFirstName
1	1	Mark
2	2	Marhia
3	3	Steven

If you want to test the encryption made on the procedure named ‘EncryptCustomerInfo’, you can use `sp_helptext` as it displays the definition of a user-defined rule.

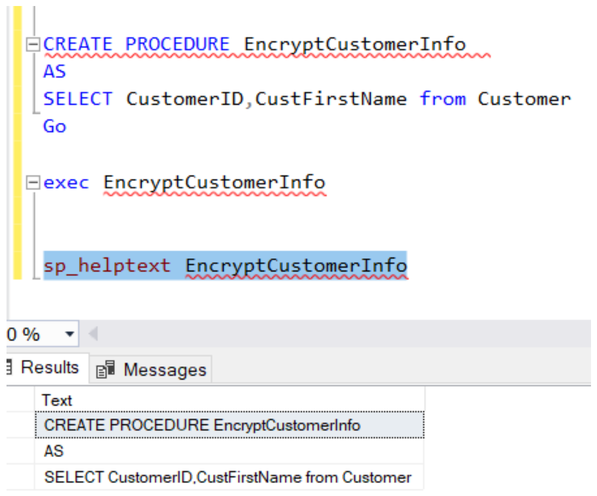
```
Exec sp_helptext 'EncryptCustomerInfo'
```



```
The text for object 'EncryptCustomerInfo' is encrypted.

Completion time: 2021-12-03T23:15:21.4113595+02:00
```

Note: When we try to view the code of the stored procedure using sp_helptext, it returns “The text for object ‘EncryptCustomerInfo is encrypted.”

If the procedure has no encryption, the sp_helptext will return the code script of the procedure as shown in the following photo.

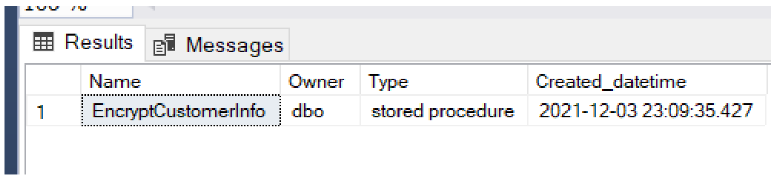


```
CREATE PROCEDURE EncryptCustomerInfo
AS
SELECT CustomerID,CustFirstName from Customer
Go

exec EncryptCustomerInfo


sp_helptext EncryptCustomerInfo
```

```
Text
CREATE PROCEDURE EncryptCustomerInfo
AS
SELECT CustomerID,CustFirstName from Customer
```

- *Also, there is* ***sp_help:***

```
Exec sp_help 'EncryptCustomerInfo'
```



```
   Name                 Owner  Type              Created_datetime
1  EncryptCustomerInfo  dbo    stored procedure  2021-12-03 23:09:35.427
```

Renaming the stored procedure

To rename a stored procedure, use system stored procedure sp_rename. The following example renames the procedure to a new name.

Example 9:

Change the name of the procedure named as "ProcedureName1" ***to*** "CustomerdetailedInfo"**.**

First, we write the procedure and execute it

```
CREATE PROCEDURE ProcedureName1
AS
Select *
FROM Customer
GO

execute ProcedureName1
```

Then, we rename the procedure and execute it with the new name

```
sp_rename 'ProcedureName1','CustomerdetailedInfo'

execute CustomerdetailedInfo
```

Note: If you tried to execute the old name of the procedure you will get an error that this object is not found in the database, to execute this procedure again you will use the new name.

Example 10: (EXTRA Question)

Create a procedure called CustomerDataCheck that check if a customer is a new customer, it will allow the user to enter his/her data into Customer table. If not, it will update his/her data with new values. And write the execution statement.

Exercise

1- Retrieve the customer ID, first name and the order ID of order that customer number 1 did.

2- Retrieve the information of order 101 and its status.

3- Once the user has entered or passed the product name (ex: tv), you should retrieve this product name and also its description. Then print them out with the alias “The name of the product is TV and its description is Smart TV”.

4- Calculate the area of the circle (A=PI *r*r) note that the Pi= 3.14

5- Hide all the information for the customer and his order using stored procedure.