SCS 2209 - DATABASE II

Instructions

- Download the ***.sql*** file in the UGVLE and import the file to ***phpmyadmin***.
- Use the ***salescompany*** database to write SQL queries for each question.
- Save your sql queries and result sets to a text document with the relevant question number.

  Make sure you have saved the text file using your index number.
- Then upload it to the UGVLE.

What is a Stored Procedure?

A stored procedure is a prepared SQL code that you can store in the database. You can pass parameters to a stored procedure to get data based on Dynamic values. A stored procedure can accept parameters, and you can set variables, write IF statements, etc within a stored procedure.

Syntax :

CREATE PROCEDURE *name_of_the_stored_procedure* (*parameter_1* **INT**)
BEGIN
    ...MySQL query goes here...
END;

If no parameters are required, the parentheses can be empty. In most cases, you will also need to surround the **CREATE PROCEDURE** statement with **DELIMITER** commands and change **END;** to **END //**.

Ex :

DELIMITER //

```
CREATE PROCEDURE name_of_the_stored_procedure (parameter_1 INT)
BEGIN
    ...MySQL query goes here...
END //

DELIMITER ;
```

Here you need to add a couple of DELIMITER commands and replace the semicolon with two forward slashes. The DELIMITER command allows us to tell MySQL to use a different delimiter.

We did this to tell MySQL to use a different delimiter while it creates the stored procedure. The reason for this is that, MySQL already recognizes the semicolon as a delimiter for marking the end of each SQL statement. Therefore, as soon as MySQL sees the first semicolon, it will interpret the delimiter as such and the stored procedure would break.

In the above example we set DELIMITER to two forward slashes (//) but this could've been anything (although, avoid using a backslash (\) as that is the escape character for MySQL). By changing the delimiter, MySQL won't try to interpret semicolons as the end of the statement. It will wait until it sees the two forward slashes.

Once we've created the stored procedure, we can use DELIMITER ; to reset the delimiter back to the semicolon.

Why Stored Procedure?

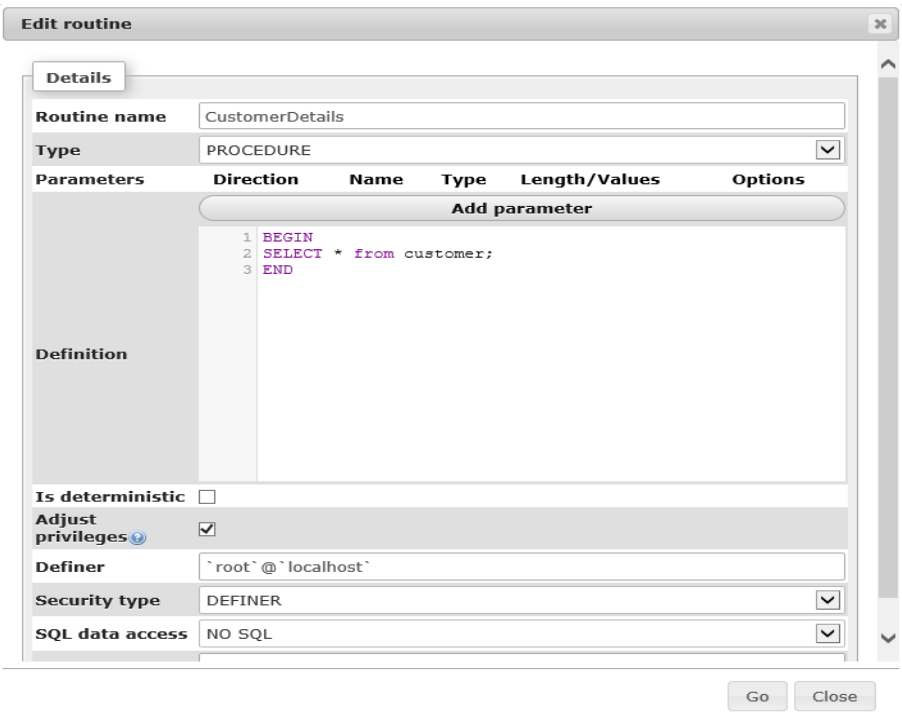
A stored procedure can be reused over and over again.

How to Create a Stored Procedure in PHPMyAdmin?

Method 01

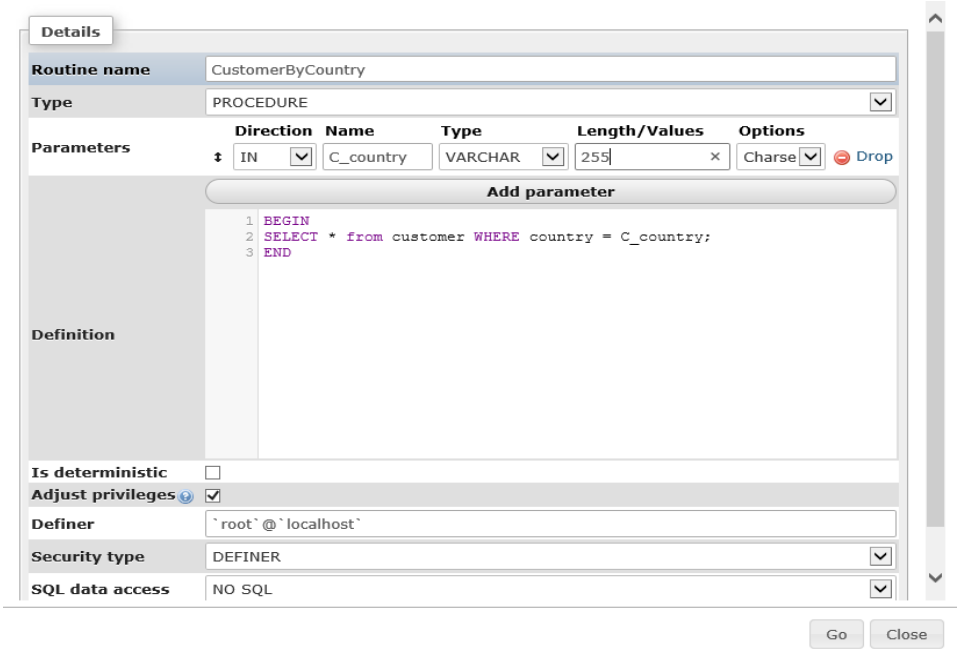
Step -1 : Open *phpMyAdmin* and select the database to create stored procedure.
Step -2 : Go to **Routines** menu & Click on **Add routine**. *phpMyAdmin* will open a pop-up.

Example **without** parameters :
Create the stored procedure *'CustomerDetails'* to get data from Customer table.



Example **with** parameters:

Create the stored procedure *'CustomerByCountry'* to get data from Customer table with respect to Customer's living Country.



Method 2 : Go to **SQL** menu and write the MySql query to *CREATE* the Stored Procedure.

How to Execute Stored Procedure from *phpMyAdmin* ?

phpMyAdmin will display the list of created Stored Procedures under the 'Procedures' sub section in your database.

Method 1 : Go to **Routines** menu and click on **Execute** link to run a specific Stored Procedure. Procedure without parameters will directly run the query and list out the data. Stored Procedures with parameters will open Pop up to add parameters, then run Procedure and get result data.

Method 2 : Go to **SQL** menu and write the MySql query to *CALL* the Stored Procedure.

Ex : *CALL* customerDetails;

CALL customerByCountry ('UK');

How to Drop a Stored Procedure ?

You can drop a Stored Procedure by using the **DROP PROCEDURE** statement.

Ex : *DROP PROCEDURE* customerDetails;

How to Alter a Stored Procedure ?

To change the body of the stored procedure, or any of its parameters, you need to *DROP* the procedure and create it again.

DROP PROCEDURE IF EXISTS *name_of_the_stored_procedure* ;

```
DELIMITER //

CREATE PROCEDURE name_of_the_stored_procedure (parameter_1 INT)
BEGIN
    ...MySQL query goes here...
END //
DELIMITER ;
```

A More Advanced Stored Procedure

Ex : Create the Stored Procedure *'CustomerRating'* to get the rating of a particular customer in the Customer table by considering the total bill amount in his/her purchase order. Criteria for the rating is as below.

TotalBillAmount	Rating
< 1000	Law rated Customer
>= 1000 and <=2000	Middle rated Customer
> 2000	Top rated Customer

Answer

```
DROP PROCEDURE IF EXISTS CustomerRating;

DELIMITER //

CREATE PROCEDURE CustomerRating(
        IN Customer_id VARCHAR(6),
        OUT Rating VARCHAR(50))

   BEGIN
        DECLARE TotalBillAmount  INT;

        SELECT
                purchase_order.TotalAmount into TotalBillAmount
        FROM
                purchase_order INNER JOIN customer ON
                purchase_order.CustomerId = customer.Id
        WHERE
                customer.Id = Customer_id;

       IF TotalBillAmount > 2000 THEN
                SET Rating = 'Top rated Customer';
       ELSEIF (TotalBillAmount <= 2000 AND TotalBillAmount >= 1000) THEN
                SET Rating = 'Middle rated Customer';
       ELSEIF (TotalBillAmount < 1000) THEN
                SET Rating = 'Law rated Customer';
        END IF;

END //
```

DELIMITER ;

The above example accepts two different modes of parameters (**IN** and **OUT**). IN is the default.

We use DECLARE *Customer_id* INT to declare a variable called *Customer_id* with a type of INT. Use a SELECT statement to look up the total billing amount in the purchase order placed by the given the Customer *ID and* assign that into our ***TotalBillAmount*** variable. SQL IF statement is used to determine the rating by placing this value into the ***Rating*** parameter (OUT parameter — the output value when called the stored procedure).

Calling a Stored Procedure with IN - OUT Parameter

For calling the stored procedure 'CustomerRating', you need to include the OUT parameter also. Since you won't know the value of OUT parameter,you need to use a variable. Then you can use a SELECT statement to find out the value of variable.

Ex: CALL CustomerRating (76, @rating);
SELECT @rating;

Exercises

Using stored procedures,

1. Find all the details of suppliers
2. Find all the details of suppliers who are from "UK"?
3. Find the phone number of the company name "Tokyo Traders"
4. Create a Stored Procedure *'UnitPriceCategory'* where the unit price of product is more than 100 as 'High', more than 50 as 'Medium' and less than 50 as 'Low'.
5. Product supplying companies in various countries are supplying different products. Assume that you need to identify which countries are leading the product supply chain.

   Based on the count of different products they are supplying , you need to apply below rating criteria.

No of products supplying	Rating Criteria
> 9	Top Leading supplier country

>= 5 and <=9	Middle Leading supplier country
< 5	Not a Leading supplier country

You should show only **10 records** as per the below result.
Hint :Create a new table to save your final result.(Results shown in the image).Create a stored procedure to find ratings and insert data into the newly created table along with the calculated rating.

id	sup_country	no_of_products	leading_status
1	USA	12	Top Leading Supplier
2	Germany	9	Middle Leading Supplier
3	Australia	8	Middle Leading Supplier
4	UK	7	Middle Leading Supplier
5	Japan	6	Middle Leading Supplier
6	Italy	5	Middle Leading Supplier
7	Sweden	5	Middle Leading Supplier
8	France	5	Middle Leading Supplier
9	Canada	4	Not a Leading Supplier
10	Norway	3	Not a Leading Supplier

6. Create the Stored Procedure *'OrderItemCount'* to get the count of items in a given customer's purchase order.

7. What are the advantages of having stored procedures and when to use stored procedures?