

Hands-on Lab: Stored Procedures



Estimated time needed: 20 minutes

Stored Procedures in SQL are a type of database object that allow you to encapsulate a series of SQL statements into a single routine. They are stored in the database data dictionary and can be invoked from an application program or from the database command interface. Stored procedures can accept input parameters and return multiple values of output parameters. They can also include control-of-flow constructs such as loops and conditional statements. Stored procedures offer several benefits including improved performance, higher productivity, ease of use, and increased scalability. They also provide a mechanism for enforcing business rules and data integrity in the database system.

Objectives

After completing this lab, you will be able to:

- Create stored procedures
- Execute stored procedures

Software Used in this Lab

In this lab, you will use [MySQL](#). MySQL is a Relational Database Management System (RDBMS) designed to efficiently store, manipulate, and retrieve data.



To complete this lab you will utilize MySQL relational database service available as part of IBM Skills Network Labs (SN Labs) Cloud IDE. SN Labs is a virtual lab environment used in this course.

Database Used in this Lab

`Mysql_learners` database has been used in this lab.

Data Used in this Lab

The data used in this lab is internal data. You will be working on the `PETSALE` table.

ID	ANIMAL	SALEPRICE
1	Cat	450.09
2	Dog	666.66
3	Parrot	50.00
4	Hamster	60.60
5	Goldfish	48.48

This lab requires you to have the `PETSALE` table populated with sample data on mysql phpadmin interface. You might have created and populated a `PETSALE` table in a previous lab.

For this lab, you need to create a database `PETS` in the phpMyAdmin interface. Download the `PETSALE-CREATE-v2.sql` script below, upload it to console under the `PETS` database. Upon execution, the script will create a new `PETSALE` table dropping any previous `PETSALE` table if exists, and will populate it with the required sample data.

- [PETSALE-CREATE-v2.sql](#)

Stored Procedure: Exercise 1

In this exercise, you will create and execute a stored procedure to read data from a table on mysql phpadmin using SQL.

1. You will create a stored procedure routine named **RETRIEVE_ALL**.

- o This **RETRIEVE_ALL** routine will contain an SQL query to retrieve all the records from the PETSALE table, so you don't need to write the same query over and over again. You just call the stored procedure routine to execute the query everytime.
- o To create the stored procedure routine, copy the code below and paste it to the textarea of the **SQL** page. Click **Go**.

```
DELIMITER //
CREATE PROCEDURE RETRIEVE_ALL()
BEGIN
    SELECT * FROM PETSALE;
END //
DELIMITER ;
```

Run SQL query/queries on database Mysql_learners: 

```
1 DELIMITER //
2
3 CREATE PROCEDURE RETRIEVE_ALL()
4
5 BEGIN
6
7     SELECT * FROM PETSALE;
8
9
10 END //
11
12 DELIMITER ;
```

Bind parameters 

[Delimiter :] Show this query here again Retain query box Rollback when finished Enable foreign key checks

 MySQL returned an empty result set (i.e. zero rows). (Query took 0.0064 seconds.)

```
CREATE PROCEDURE RETRIEVE_ALL() BEGIN SELECT * FROM PETSALE; END
```

2. To call the **RETRIEVE_ALL** routine, open another **SQL** tab by clicking **Open in new Tab**

The screenshot shows the phpMyAdmin interface. The left sidebar lists databases and tables under the HR schema. The main area is the SQL tab, displaying a query: `1 SELECT * FROM `EMPLOYEES``. A context menu is open over this query, with the first option, "Open link in new tab", highlighted with a red box.

Delete the default line which appears so that you will get a blank window.

Copy the code below and paste it to the textarea of the **SQL** page. Click **Go**.

```
CALL RETRIEVE_ALL;
```

```
11 CALL RETRIEVE_ALL;
```

Bind parameters 

Delimiter Show this query here again Retain query box Rollback when finished Enable foreign key checks

[Hide query box](#)

 Showing rows 0 - 4 (5 total, Query took 0.0010 seconds.)

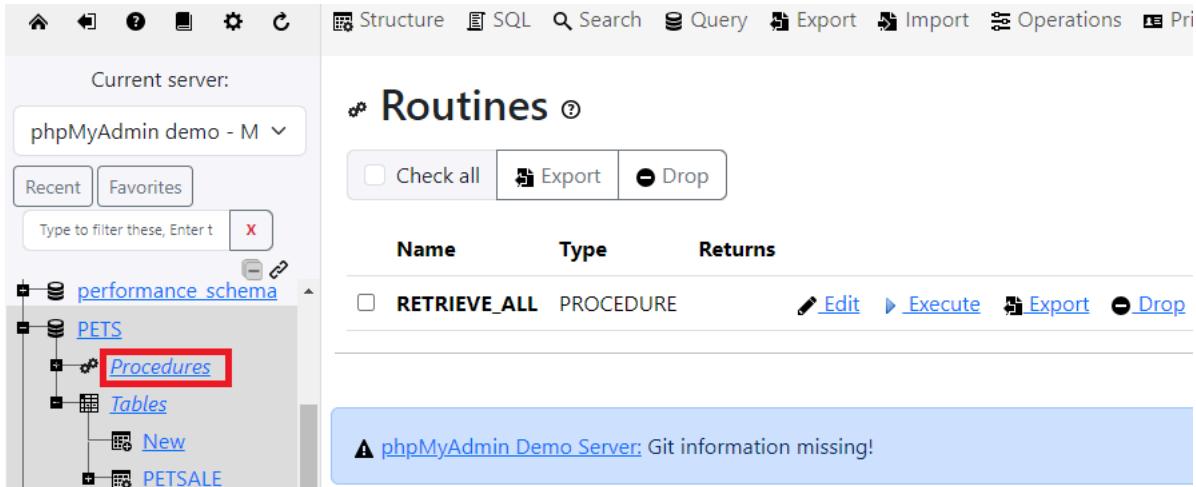
`CALL RETRIEVE_ALL`

Show all | Number of rows: Filter rows:

Options

	ANIMAL	SALEPRICE	SALEDATE	QUANTITY
1	Cat	450.09	2018-05-29	9
2	Dog	666.66	2018-06-01	3
3	Parrot	50.00	2018-06-04	2
4	Hamster	60.60	2018-06-11	6
5	Goldfish	48.48	2018-06-14	24

3. You can view the created stored procedure routine RETRIEVE_ALL. On the left panel, expand the PETS database option and click on **Procedures** to view the procedure.



The screenshot shows the phpMyAdmin interface. The left sidebar shows a tree structure with 'Current server: phpMyAdmin demo - M'. Under the 'PETS' database, the 'Procedures' node is highlighted with a red box. The main area is titled 'Routines' and lists a single procedure:

Name	Type	Returns
RETRIEVE_ALL	PROCEDURE	Edit Execute Export Drop

A message box at the bottom states: **phpMyAdmin Demo Server:** Git information missing!

4. If you wish to drop the stored procedure routine RETRIEVE_ALL, copy the code below and paste it to the textarea of the SQL page. Click Go.

```
DROP PROCEDURE RETRIEVE_ALL;
CALL RETRIEVE_ALL;
```

The screenshot shows the MySQL Workbench interface. In the top navigation bar, the 'SQL' tab is selected. Below the tabs, there is a code editor containing the following SQL code:

```

1
2     DROP PROCEDURE RETRIEVE_ALL;
3
4     CALL RETRIEVE_ALL;
5
6

```

Below the code editor are several buttons: 'Clear', 'Format', 'Get auto-saved query', and a checkbox for 'Bind parameters'. Underneath these are additional configuration options: '[Delimiter :]', checkboxes for 'Show this query here again', 'Retain query box', 'Rollback when finished', and 'Enable foreign key checks' (which is checked), and a link 'Get auto-saved query'.

A red box highlights the 'Error' section below the configuration. It contains the following text:

Error

SQL query: [Copy](#)

CALL RETRIEVE_ALL

MySQL said: [?](#)

#1305 - PROCEDURE Mysql_learners.RETRIEVE_ALL does not exist

Stored Procedure: Exercise 2

In this exercise, you will create and execute a stored procedure to write/modify data in a table on MySQL using SQL.

You will create a stored procedure routine named **UPDATE_SALEPRICE** with parameters **Animal_ID** and **Animal_Health**.

- This **UPDATE_SALEPRICE** routine will contain SQL queries to update the sale price of the animals in the PETSALE table depending on their health conditions, **BAD** or **WORSE**.
- This procedure routine will take animal ID and health condition as parameters which will be used to update the sale price of animal in the PETSALE table by an amount depending on their health condition. Suppose that:
 - For animal with ID XX having BAD health condition, the sale price will be reduced further by 25%.
 - For animal with ID YY having WORSE health condition, the sale price will be reduced further by 50%.
 - For animal with ID ZZ having other health condition, the sale price won't change.
- To create the stored procedure routine, copy the code below and paste it to the textarea of the **SQL** page. Click **Go**.

```

DELIMITER @@
CREATE PROCEDURE UPDATE_SALEPRICE (IN Animal_ID INTEGER, IN Animal_Health VARCHAR(5))
BEGIN
  IF Animal_Health = 'BAD' THEN
    UPDATE PETSALE
    SET SALEPRICE = SALEPRICE - (SALEPRICE * 0.25)
    WHERE ID = Animal_ID;
  ELSEIF Animal_Health = 'WORSE' THEN
    UPDATE PETSALE
    SET SALEPRICE = SALEPRICE - (SALEPRICE * 0.5)
    WHERE ID = Animal_ID;
  ELSE
    UPDATE PETSALE
    SET SALEPRICE = SALEPRICE
    WHERE ID = Animal_ID;
  END IF;
END @@
DELIMITER ;

```

Run SQL query/queries on database **Mysql_learners**: 

```
15
16     ELSE
17         UPDATE PETSALE
18             SET SALEPRICE = SALEPRICE
19             WHERE ID = Animal_ID;
20
21     END IF;
22
23 END @
24
25 DELIMITER ;
26
```

Bind parameters 

[Delimiter] Show this query here again Retain query box Rollback when finished Enable foreign key checks

Hide query box

 MySQL returned an empty result set (i.e. zero rows). (Query took 0.0214 seconds.)

```
CREATE PROCEDURE UPDATE_SALEPRICE ( IN Animal_ID INTEGER, IN Animal_Health VARCHAR(5) ) BEGIN IF Animal_Health = 'BAD' THEN SALEPRICE * 0.25 END IF; ELSEIF Animal_Health = 'WORSE' THEN UPDATE PETSALE SET SALEPRICE = SALEPRICE * 0.25; ELSE UPDATE PETSALE SET SALEPRICE = SALEPRICE WHERE ID = Animal_ID; END IF; END
```

1. Let's call the UPDATE_SALEPRICE routine. We want to update the sale price of animal with ID **1** having **BAD** health condition in the PETSALE table. open another **SQL** tab by clicking **Open in new Tab**

The screenshot shows the phpMyAdmin interface. The left sidebar lists databases and tables under the HR schema. The main area is the SQL tab, showing a query editor with the following code:

```
1 SELECT * FROM `EMPLOYEES`
```

Below the query editor are several buttons: SELECT*, SELECT, INSERT, UPDATE, DELETE, Clear, Format, and Get au. There is also a checkbox for Bind parameters. At the bottom, there is a Delimiter input field set to ';' and three checkboxes: Show this query here again, Retain query box, and Rollback when finish.

A context menu is open over the query editor, with the 'Open link in new tab' option highlighted by a red box. Other options in the menu include Open link in new window, Open link in incognito window, Save link as..., Copy link address, and Inspect.

Delete the default line which appears so that you will get a blank window.

Copy the code below and paste it to the textarea of the SQL page. Click Go.

Note if you have dropped RETREIVE_ALL procedure rerun the creation script of that procedure before executing these lines.

```
CALL RETRIEVE_ALL;
CALL UPDATE_SALEPRICE(1, 'BAD');
CALL RETRIEVE_ALL;
```

Showing rows 0 - 4 (5 total, Query took 0.0007 seconds.)

CALL RETRIEVE_ALL

Show all | Number of rows: 25 ▾ Filter rows: Search this table

+ Options

ID	ANIMAL	SALEPRICE	SALEDATE	QUANTITY
1	Cat	450.09	2018-05-29	9
2	Dog	666.66	2018-06-01	3
3	Parrot	50.00	2018-06-04	2
4	Hamster	60.60	2018-06-11	6
5	Goldfish	48.48	2018-06-14	24

Note: #126

Showing r

CALL RETRIEVE_ALL

Show a

+ Options

ID	ANIMAL
1	Cat
2	Dog
3	Parrot
4	Hamster
5	Goldfish

2. Let's call the UPDATE_SALEPRICE routine once again. We want to update the sale price of animal with ID 3 having WORSE health condition in the PETSALE table. copy the code below and paste it to the textarea of the SQL page. Click Go. You will have all the records retrieved from the PETSALE table.

```
CALL RETRIEVE_ALL;
CALL UPDATE_SALEPRICE(3, 'WORSE');
CALL RETRIEVE_ALL;
```

CALL RETRIEVE_ALL

Showing rows 0 - 4 (5 total,

CALL RETRIEVE_ALL

Show all | Number of rows: 25 ▾ Filter rows: Search this table

Options

ID	ANIMAL	SALEPRICE	SALEDATE	QUANTITY
1	Cat	337.57	2018-05-29	9
2	Dog	666.66	2018-06-01	3
3	Parrot	50.00	2018-06-04	2
4	Hamster	60.60	2018-06-11	6
5	Goldfish	48.48	2018-06-14	24

CALL RETRIEVE_ALL

Show all | Number of r

Options

ID	ANIMAL	SALEPRICE
1	Cat	337.57
2	Dog	666.66
3	Parrot	25.00
4	Hamster	60.60
5	Goldfish	48.48

Query results operations

3. You can view the created stored procedure routine UPDATE_SALEPRICE. Click on the **Routines** and view the procedure.

The screenshot shows the MySQL Workbench interface with the 'Routines' tab selected. The table lists two stored procedures:

Name	Action	Type	Returns
<input type="checkbox"/> RETRIEVE_ALL	Edit Execute Export Drop	PROCEDURE	
<input type="checkbox"/> UPDATE_SALEPRICE	Edit Execute Export Drop	PROCEDURE	

Below the table are buttons for 'Check all', 'With selected:', 'Export', and 'Drop'. A 'New' button is also present.

4. If you wish to drop the stored procedure routine UPDATE_SALEPRICE, copy the code below and paste it to the textarea of the SQL page. Click Go.

```
DROP PROCEDURE UPDATE_SALEPRICE;
CALL UPDATE_SALEPRICE;
```

The screenshot shows the SQL editor with the following code:

```
7
8
9 DROP PROCEDURE UPDATE_SALEPRICE;
10
11 CALL UPDATE_SALEPRICE;
```

Below the code are buttons for 'Clear', 'Format', and 'Get auto-saved query'. There is also a checkbox for 'Bind parameters'.

At the bottom, there are options for 'Delimiter' (set to ;), 'Show this query here again' (unchecked), 'Retain query box' (unchecked), 'Rollback when finished' (unchecked), and 'Enable foreign key checks' (checked).

[Hide query box](#)

Error

SQL query: [Copy](#)

```
DROP PROCEDURE UPDATE_SALEPRICE
```

MySQL said: [?](#)

```
#1305 - PROCEDURE Mysql_learners.UPDATE_SALEPRICE does not exist
```

Conclusion

Congratulations! You have completed this lab on creating stored procedures in MySQL.

You are now able to:

- Write a stored procedure as per requirement
- Call or Execute a stored procedure
- Drop a stored procedure once its utility is over

Author(s)

[Lakshmi Holla](#)

[Malika Singla](#)

[Abhishek Gagneja](#)

© IBM Corporation 2023. All rights reserved.