DHA SUFFA UNIVERSITY
Department of Computer Science

CS-2003 Database Systems
Spring 2022

LAB 10 Stored Procedures

OBJECTIVE(S)

- Learn about stored procedures.
- Learn about conditional statements.
- Learn about loops.

STORED PROCEDURES

A stored procedure is a subroutine (like a subprogram in a regular computing language) stored in a database. It has a name, a parameter list, and SQL statement(s). Stored procedures are supported by almost all relational database systems.

Advantages of Stored Procedures

Improved performance — Once created, stored procedures are compiled and stored in the database. MySQL, however, compiles the stored procedure on demand and stores it in its cache, maintaining its own stored procedure cache for every single connection. If an application uses a stored procedure multiple times in a single connection, the compiled version is used, otherwise, the stored procedure works like a regular query.

Reduction in network traffic — Repetitive tasks, especially those involving multiple lengthy SQL statements can be replaced with a stored procedure. Hence, instead of sending multiple SQL queries, the application can now perform the same task with a single call using the name and parameters of a stored procedure stored on the serve thereby reducing the traffic between the application and database server.

Reusability and portability — Once prepared, a SQL stored procedure will run on any platform that MySQL runs on without requiring any additional support.

Disadvantages of Stored Procedures

Increased memory usage — When using many stored procedures, the memory usage of every connection using those stored procedures will increase substantially.

Difficult to debug and maintain — Developing and maintaining stored procedures require a specialized skill set the lack of which may lead to problems during both application development and maintenance.

USING STORED PROCEDURES

Creation

The CREATE PROCEDURE command is used to create a new stored procedure.

In order to write stored procedures in MySQL, we first need to redefine the delimiter (;) temporarily. This is done to pass the stored procedure to the server as a whole rather than letting the mysql tool interpret each statement at a time.

As an example, consider the following stored procedure displays all the data of the Students table.

```
DELIMITER $$
CREATE PROCEDURE GetAllStudents()
    BEGIN
        SELECT * FROM students;
    END $$
DELIMITER ;
```

Calling

The CALL keyword followed by the procedure name is used to call a stored procedure. For example:

- **CALL** GetAllStudents();

Using Variables

Variables are named data objects whose value can change during the stored procedure execution. They must be declared before use.

To declare a variable inside a stored procedure, we make use of the following syntax:

- **DECLARE** variable_name datatype(size) **DEFAULT** default_value;

To assign a value to the variable, we make use of the **SET** statement:

- **SET** variable_name = new_value;

Finally, to assign the result of a query to the variable, we make use of the **SELECT INTO** statement.

- **SELECT** col_name **INTO** variable_name **FROM** tb_name;

For example:

```
DECLARE total_students int(20) DEFAULT 0;
SET total_students = 10;
SELECT COUNT(*) INTO total_students FROM students;
```

Using Parameters

Parameters make the stored procedure more flexible and useful. To define a parameter in a stored procedure, we make use of the following syntax.

- **MODE** param1_name param1_type(param1_size),
    MODE param2_name param2_type(param2_size)

In MySQL, a parameter can have three modes:

IN — This is the default mode. When using the **IN** mode, the calling program has to pass an argument to the stored procedure. The value of an **IN** parameter is protected i.e. changing the value of the **IN** parameter inside the stored procedure has no effect on the original value of the parameter which is retained after the procedure ends. In other words, the stored procedure only works on a copy of the **IN** parameter.

OUT — The value of the **OUT** parameter can be changed inside the stored procedure and the new value is then passed back to the calling program. Note that the stored procedure cannot access the initial value of the **OUT** parameter when it starts.

INOUT — An **INOUT** parameter is a combination of both **IN** and **OUT** parameters possessing all the properties of both.

Note: OUT and INOUT parameters are generally used when we want to return more than one value from a stored procedure since, by default, stored procedures return only one value.

For example:

```
CREATE PROCEDURE SetCount(INOUT count int, IN inc int)
SET count = count + inc;

DELIMITER //
CREATE PROCEDURE CountStudentsDept(IN deptname varchar(100),
                                   OUT totalstudents int)
    BEGIN
        SELECT COUNT(*) INTO totalstudents
            FROM students
            WHERE deptid =
            (SELECT id FROM dept WHERE name = deptname);
    END//
    DELIMITER ;
```

To execute the above procedures, we would use:

```
CALL CountStudentsDept("CS", @total);
SELECT @total AS "Total Students";

SET @counter = 1;
CALL setcount(@counter, 1);
CALL setcount(@counter, 5);
```

DELETING PROCEDURES

To display the procedures created in a particular database, we use the following command:

- **SHOW PROCEDURE STATUS WHERE DB =** "db_name";

Finally, to delete a procedure, we use the following command:

- **DROP PROCEDURE** procedure_name;

TASK

- Write a procedure to display the number of students that have passed or failed in each department.
  [Hint: Use 2.5 GPA as threshold]
- Write a single procedure to display the old salary of any faculty member. Using the same procedure, update the salary of employee using a given bonus percentage. Also, display the updated salary.

CONDITONAL STATEMENTS

Conditional statements, much like in conventional programming languages, are used to alter the flow of the stored procedure based on certain conditions or values of expressions. . To form an expression in MySQL, we can combine literals, variables, operators and even functions. An expression can result TRUE, FALSE, or NULL. MySQL allows three types of condition statements in stored procedures.

IF Statement

The syntax for the IF statement is:

-
  ```
  IF expression THEN
          statement(s);
  END IF;
  ```

-
  ```
  IF expression THEN
          statement(s);

  ELSE
          statement(s);
  END IF;
  ```

-
  ```
  IF expression THEN
          statement(s);

  ELSEIF expression THEN
          statement(s);

  ELSEIF expression THEN
          statement(s);

  ELSE
          statement(s);
  END IF;
  ```

Consider the following example:

```
DELIMITER $$
CREATE PROCEDURE SalaryStatus(IN fac_id int, OUT salary_status varchar(255))
        BEGIN
                DECLARE current_salary float(10,2);
                DECLARE average_salary float(10,2);

                SELECT avg(salary) INTO average_salary FROM faculty;
                SELECT salary INTO current_salary FROM faculty
                        WHERE id = fac_id;

                IF current_salary < average_salary THEN
                        SET salary_status = “Less than average salary”;
                ELSEIF current_salary = average_salary THEN
                        SET salary_status = “Equal to average salary”;
                ELSEIF current_salary > average_salary THEN
                        SET salary_status = “Greater than average salary”;
                END IF;

        END$$
        DELIMITER ;
```

Simple CASE Statement

The statement is an alternative conditional statement to the IF statement. It has an added benefit of making the code more readable and efficient. A simple CASE statement can be used to check the value of an expression against a set of unique values.
The ELSE clause is optional and is executed if none of the WHEN expressions match the CASE expression. If, however, the ELSE clause is omitted and no match is found, MySQL will raise an error.

```
• CASE case_expression
        WHEN expression1 THEN statement(s)
        WHEN expression2 THEN statement(s)
        ELSE statement(s);
  END CASE;
```

Searched CASE Statement

The simple CASE statement only allows matching a value of an expression against a set of distinct values. In order to perform more complex matches such as ranges, we make use of the searched CASE statement. The searched CASE statement is equivalent to the IF statement, however, its construct is much more readable.

```
• CASE
        WHEN condition1 THEN statement(s)
        WHEN condtion2 THEN statement(s)
        ELSE statement(s);
  END CASE;
```

MySQL evaluates each condition in the WHEN clause until a condition is reached whose value is TRUE thereby executing the corresponding THEN statement.

TASK

- Write a procedure to display whether a student has passed or failed.
  (Hint: Use 2.5 GPA as threshold.)
- Repeat the above task using case statement.

LOOP STATEMENTS

Loop statements are used to run a block of SQL code repeatedly based on a condition. MySQL provides three loop statements: **WHILE**, **REPEAT**, and **LOOP**.

WHILE

The WHILE loop, also called the pretest loop, checks the expression at the beginning of each iteration. If the expression evaluates to TRUE, the statements between the WHILE and END WHILE will be executed until the expression evaluates to FALSE.

```
• WHILE expression DO
          statement(s);
  END WHILE;
```

REPEAT

The REPEAT loop, also called the post-test loop, evaluates the expression after the execution of the statements. If the expression evaluates to FALSE, the loop is executed repeatedly until the expression evaluates to TRUE. For this reason, the statements within a REPEAT loop will be executed at least once.

```
• REPEAT
          statement(s);
          UNTIL expression
  END REPEAT;
```

LOOP

LOOP statement executes a block of code repeatedly with an additional flexibility of using a loop label. The expression to end or continue the loop is evaluated within the statements inside the loop block.

LEAVE — The LEAVE statement allows us the exit the loop immediately without waiting for checking the condition. This is analogous to the **break** statement in conventional programming languages.

ITERATE — The ITERATE statement allows us to skip the entire code under it and start a new iteration. This is analogous to the **continue** statement in conventional programming languages.

- loop_label: **LOOP**
  ```
  loop_label: LOOP
      statement(s);
      LEAVE loop_label;
      statement(s);
      ITERATE loop_label;
  END LOOP;
  ```

TASK

- Modify the pass-fail procedure to include a loop that performs the operation on all the students.

LAB ASSIGNMENT

1. Write a procedure to display the name, designation, and salary of the highest paid employee. Use a subquery to get the highest paid employee.
2. Write a procedure to display the details of all the employees whose salary is more than the salary specified as input to the procedure.
3. Write a procedure that gives the average of the minimum salaries of all the departments. The average salary should be returned in a variable that is accessible outside the procedure.
4. Create a procedure that takes employee ID as input and returns the bonus percentage for the employee as output using the following criteria:
   Salary more than 70% of maximum salary — 5% bonus
   Salary more than 50% but less than/equal to 70% of maximum salary — 15% bonus
   Salary more than 20% but less than/equal to 50% of maximum salary — 25% bonus
   Salary less than or equal to 20% of maximum salary — 35% bonus
   [Hint: Use if statements]
5. Repeat the above task using case statement.
6. Modify your procedure to include a loop that performs the above task for all the employees automatically.

SUBMISSION GUIDELINES

- Take a screenshot of each task. Ensure that all screenshots have a white background and black text. You can alter the background and text colors through the properties of the MySQL command line client.
- Place all the screenshots in a single word file labeled with Roll No and Lab No. e.g. **‘cs181xxx_Lab10’**
- Convert the file into PDF.
- Submit the PDF file at LMS
- **-100%** policies for plagiarism.