Lab 6

Stored procedure

- ✓ A stored procedure is a named collection of SQL statements that are precompiled and stored in a database.
- ✓ If the user has an SQL query that you write over and over again, keep it as a stored procedure and execute it.
- ✓ Users can also pass parameters to a stored procedure so that the stored procedure can act based on the parameter value that is given.
- ✓ Based on the statements in the procedure and the parameters we pass, it can perform one or multiple DML operations on the database, and return value, if any.
- ✓ Thus, it allows us to pass the same statements multiple times, thereby, enabling reusability.

Advantages

- ✓ **Reusability:** Once a stored procedure is created, it can be called multiple times from different parts of an application or by multiple users.
- ✓ **Improved performance**: Since stored procedures are precompiled and stored in the database, they can execute faster than sending individual SQL statements from an application to the database server. This is because the database server doesn't have to re-parse and optimize the code each time it is executed.
- ✓ **Reduced network traffic:** The server only passes the procedure name and parameter instead of the whole query, reducing network traffic.
- ✓ **Easy to modify:** WE can easily change the statements in a stored procedure as per necessary.
- ✓ **Security:** Stored procedures can provide an additional layer of security by allowing access to the underlying data through the procedure while restricting direct access to the tables.
- ✓ **Modularity and encapsulation:** Stored procedures enable the modularization and encapsulation of database logic, making it easier to maintain and update the database code.
- ✓ **Parameterization**: Stored procedures can accept input parameters, allowing you to pass values into the procedure at runtime. These parameters can be used within the SQL statements to make the procedure more flexible and reusable

Creating stored procedure

To create a stored procedure in **MySQL**, we can use the following syntax:

```
DELIMITER //
CREATE PROCEDURE procedure_name(parameter1 datatype, parameter2 datatype, ...)
BEGIN
  ----Statements using the input parametes
END //
DELIMITER ;
```

Note:

- ✓ **DELIMITER //** is used to change the delimiter temporarily so that you can use the semicolon ; within the procedure body without ending the entire statement prematurely.
- ✓ **DELIMITER ;** sets the delimiter back to the default semicolon ;

Executing stored procedure

To execute a stored procedure in MySQL, you can use the CALL statement followed by the name of the procedure and list of parameters if any. The syntax is as follows:

```
CALL procedure_name(parameter_list);
```

Creating stored procedure without parameters

Example:

```
DELIMITER //
CREATE PROCEDURE  getallEmployee ()
BEGIN
SELECT * FROM employee;
END //
DELIMITER ;
```

Now, we can execute the above stored procedure as follows

```
CALL getallEmployee();
```

Creating parameterized stored procedure

In SQL, a parameterized procedure is a type of stored procedure that can accept input parameters. These parameters can be used to customize the behavior of the procedure and perform operations based on the input values provided.

To create a parameterized stored procedure in MySQL, we can define input parameters within the procedure definition.

Example:

```
DELIMITER //
CREATE PROCEDURE getdepartmentEmployee (dept varchar(30))
BEGIN
SELECT *
FROM employee
WHERE department=dept;
END //
DELIMITER ;
```

To call this stored procedure, you can use the CALL statement and pass the parameter value:

```
CALL getdepartmentEmployee('civil');
```

Drop procedure

We can use the DROP PROCEDURE statement followed by the name of the procedure.

Here's the syntax:

```
DROP PROCEDURE procedure_name;
```

Example:

DROP PROCEDURE getdepartmentEmployee;