**PROCEDURES**

**EXPT NO: 9 DATE: 6/12/22**

**AIM:**  To implement procedures in SQL

**THEORY**

A procedure (often called a stored procedure) is a**collection of pre-compiled SQL statements** stored inside the database. It is a subroutine or a subprogram in the regular computing language.

**A procedure always contains a name, parameter lists, and SQL statements.** We can invoke the procedures by using triggers, other procedures and applications such as [Java](https://www.javatpoint.com/java-tutorial), [Python](https://www.javatpoint.com/python-tutorial), [PHP](https://www.javatpoint.com/php-tutorial), etc. It was first introduced in MySQL **version 5.** Presently, it can be supported by almost all relational database systems.

If we consider the enterprise application, we always need to perform specific tasks such as database clean up, processing payroll, and many more on the database regularly. Such tasks involve multiple [SQL](https://www.javatpoint.com/sql-tutorial) statements for executing each task. This process might easy if we group these tasks into a single task. We can fulfill this requirement in [MySQL](https://www.javatpoint.com/mysql-tutorial) by creating a stored procedure in our database.

A procedure is called a **recursive stored procedure** when it calls itself. Most database systems support recursive stored procedures but, it is not supported well in MySQL.

Parameter Types:

1. **IN parameter**

It is the default mode. It takes a parameter as input, such as an attribute. When we define it, the calling program has to pass an argument to the stored procedure. This parameter's value is always protected.

1. **OUT parameters**

It is used to pass a parameter as output. Its value can be changed inside the stored procedure, and the changed (new) value is passed back to the calling program. It is noted that a procedure cannot access the OUT parameter's initial value when it starts.

1. **INOUT parameters**

It is a combination of IN and OUT parameters. It means the calling program can pass the argument, and the procedure can modify the INOUT parameter, and then passes the new value back to the calling program.

Syntax:

To create procedures:

DELIMITER &&

**CREATE** **PROCEDURE** procedure\_name [[IN | **OUT** | INOUT] parameter\_name datatype [, parameter datatype]) ]

**BEGIN**

    Declaration\_section

    Executable\_section

**END** &&

DELIMITER ;

To call procedure:

CALL procedure\_name ( parameter(s))

To delete procedure:

**DROP** **PROCEDURE** [ IF EXISTS ] procedure\_name;

**QUERIES**

1) Write a procedure to get details of all books.

2) Write a procedure which accepts &#39;bookid&#39; as parameter and display the details.

3) Write a procedure which accepts &#39;cardno&#39; as parameter and display the name,addr,phone no and the number of books loaned out to the borrower and number of books not yet returned.

4) Write procedure which accepts cardno as an input and finds the number of books

borrowed and not returned by the borrower(call this as cnt) (even after the 1 month

period is over). If cnt &gt; 5 delete the borrower from the database.(he is not a

cardholder of the library) else cnt &gt;2 display a warning message to the borrower.

5) Write a procedure which executes the same task as in procedure 4 but for all the

borrowers in the database.

**CONCLUSION**

The concept of procedures in SQL was studied.