|  |
| --- |
|  |
|  |

**Stored Procedures** are segments of declared SQL statements stored inside the database catalog. They increase performance, reduce network traffic, provide code reusability, and security. Stored procedures can have any combinations of input, output and input/output parameters. Clients send the definitions of a procedure or function to the server, which stores it for later use. Clients then utilize it when necessary to cause SQL operations to be performed to produce values.

A construct of Stored Procedures, **Parameters** are a default type. If not declared they are just considered as an IN parameter. IN parameters pass to the procedure then can be modified and once the value is modified it will not be visible when the procedure returns. If the value of the IN parameter is changed in the stored procedure, the original value is kept after the stored procedure ends. This only provides input to parameters, which provides values to the stored procedures.

*Example*: Unfinished Code

A construct of Stored Procedures, **Out** is the opposite from the IN parameter. The OUT parameter is available in the stored procedure. The default value of OUT is NULL. The value can change inside the stored procedure and its new value is passed back to the calling program. This only provides output parameters, which returns values from the stored procedure to the calling program.

*Example*: Unfinished Code

A construct of Stored Procedures, **INOUT** is a combination of IN and OUT parameters. The calling program can pass the argument and the stored procedure can modify and pass a new value back to the calling program. This provides both output and input parameters, which provide values to and from stored procedures. The starting value of OUT parameter is NULL.

*Example*: Unfinished Code

A construct of Stored Procedures, **IN** is the default mode. The value of IN is always retained after the stored procedure ends. Input parameters add flexability to stored procedures.

Example:

Delimiter //

CREATE procedure find\_emp\_last\_name(IN name varchar(200))

SELECT emp\_no, last\_name from employees where last\_name = name;

END;

END //

CALL find\_emp\_last\_name( ‘NAME’);

A construct of Stored Procedures, **IF allows** you to execute a set of SQL statements based on certain values of an expression. If the expression is true, then the code will execute, otherwise it will move on to the next statement follow0

ing “end if”.

*Example*: Unfinished Code

A construct of Stored Procedures, **Loop** can fetch result sets in a server and loop through data one by one and execute the stored procedure.Loops allow statements to execute a section of code repeatedly based on a condition.

*Example*: Unfinished Code

**References:**

<https://dev.mysql.com/doc/connector-net/en/connector-net-tutorials-stored-procedures.html>

<https://code.tutsplus.com/articles/an-introduction-to-stored-procedures-in-mysql-5--net-17843>

<https://www.w3resource.com/mysql/mysql-procedure.php>