**Stored procedures**

IN parameter : to send input to stored procedure

IN – is the default mode. When you define an IN parameter in a stored procedure, the calling program has to pass an argument to the stored procedure. In addition, the value of an IN parameter is protected. It means that even the value of the IN parameter is changed inside the stored procedure, its original value is retained after the stored procedure ends. In other words, the stored procedure only works on the copy of the IN parameter.

Example : getting list of offices situated in a country. Country name is sent as parameter.

**Creating table office :**

create table office

(officeCode int, city text, adresss text, state text, country text,postal\_code int);

**Inserting sample data into office table :**

INSERT INTO `office` VALUES (11,'bangalore','nagavara','karnataka','india',560010),(12,'moscow','st.peter road','north','russia',199223);

**Creating stored procedure :**

DELIMITER //

create procedure getOfficeCountry(IN countryName varchar(255))

begin

select \*

from office

where country = countryName;

End //

DELIMITER ;

**Calling stored procedure :**

call getOfficeCountry("india");

--------------------------------------------------------------------------------------------------------------------------

**OUT parameter**

: to store the output value

the value of an OUT parameter can be changed inside the stored procedure and its new value is passed back to the calling program. Notice that the stored procedure cannot access the initial value of the OUT parameter when it starts.

Example : to count number of records whose status is shipped and store to OUT parameter

Create a table orders :

*create table orders(orderId int, status text, product text);*

Insert sample data with status shipped

*insert into orders(orderId, status, product) values (1,"shipped","product1");*

*insert into orders(orderId, status, product) values (2,"shipped","product2");*

*insert into orders(orderId, status, product) values (3,"shipped","product3");*

*insert into orders(orderId, status, product) values (4,"shipped","product5");*

*insert into orders(orderId, status, product) values (5,"shipped","product4");*

*insert into orders(orderId, status, product) values (6,"shipped","product6");*

*insert into orders(orderId, status, product) values (7,"shipped","product7");*

*insert into orders(orderId, status, product) values (8,"shipped","product8");*

Creating stored procedure to count status

*DELIMITER $$*

*create procedure CountStatus(IN orderStatus varchar(25),*

*OUT total INT)*

*BEGIN*

*select count(status)*

*into total*

*from orders*

*where status = orderStatus;*

*END $$*

*DELIMITER ;*

To check the result :

*call CountStatus("shipped",@total);*

*select @total*

*-----------------------------------------------------------------------------------------------------------------------*

**INOUT parameter example**

an INOUT parameter is the combination of IN and OUT parameters. It means that the calling program may pass the argument, and the stored procedure can modify the INOUT parameter and pass the new value back to the calling program.

Example : sending default value of count variable and getting updated value

**Creating stored procedure**

DELIMITER $$

CREATE PROCEDURE set\_counter(INOUT count INT(4), IN inc INT(4))

BEGIN

SET count = count + inc;

END $$

DELIMITER ;

How it works.

* The set\_counter stored procedure accepts one INOUT parameter ( count ) and one IN parameter ( inc ).
* Inside the stored procedure, we increase the counter ( count ) by the value of the inc parameter.

See how we call the set\_counter stored procedure:

SET @counter = 1;

CALL set\_counter(@counter,1); // -- sending(1,1)------>2

CALL set\_counter(@counter,1); // -- sending(2,1)------>3

CALL set\_counter(@counter,5); // -- sending(3,5)------>8

SELECT @counter; // --> 8