

DELIMITER:

a stored procedure contains multiple statements separated by semicolons (;). To compile the whole stored procedure as a single compound statement, you need to temporarily change the delimiter from the semicolon (;) to another delimiter such as \$\$ or //

Lecture 9: Advance SQL II (Stored Procedure)

By Sumayyea Salahuddin

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.

So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

Procedure:

01. **delimiter \$\$**
02. **create procedure** p2()
03. **begin**
04. select 'Testing Procedure' as Title;
05. **end\$\$**
06. **delimiter ;**

Detail: A simple procedure by name of p2 has been created. On execution, it shows **Title** as Field Name and **Testing Procedure** as its instance value. Initially delimiter has been changed to \$\$, once procedure definition is complete, it has been reset to default delimiter. This setting and resetting of delimiter is always practiced whenever new procedure is defined.

Procedure:

01. set @Name = 'Ali';
02. select @Name;
- 03.
04. **delimiter \$\$**
05. **create procedure** usp_in(**in p varchar(10)**)
06. **begin**
07. **set @Name = p;**
08. **end \$\$**
09. **delimiter ;**
- 10.
11. call usp_in('Zahid');
12. select @Name;

Detail: This example shows how a procedure can be used to set the global variables. Initially, a global variable @Name is set to some value (e.g. Ali here) and its contents are displayed. Then a procedure **usp_in** taking one IN-type varchar argument 'p' is defined. Inside this procedure, global variable is set to p. Last couple of lines shows calling this procedure and displaying global variable contents.

Procedure:

01. set @Name = 'Ali';
02. select @Name;
- 03.
04. **delimiter \$\$**
05. **create procedure** usp_out(out p varchar(100))
06. **begin**

```

07. set p = 'Zahid';
08. end$$
09. delimiter ;
10.
11. call usp_out(@Name);
12. select @Name;

```

Detail: This example shows how a procedure can be used to retrieve output value. Initially, a global variable @Name is set to some value (e.g. Ali here) and its contents are displayed. Then a procedure **usp_out** taking one OUT-type varchar argument 'p' is defined. Inside this procedure, p is set to some value (e.g. Zahid here). To call procedure having OUT type argument, it is compulsory to **provide variable name instead of value otherwise error will be generated**. So, @Name is provided as input argument during function call and results are displayed.

Procedure:

```

01. delimiter $$
02. create procedure usp_inout(inout p int)
03. begin
04. set p = p+2;
05. end$$
06. delimiter ;
07.
08. set @param_1 = 5;
09. call usp_inout(@param_1);
10. select @param_1;

```

Detail: This example shows how a procedure can be used to both set and retrieve a value. Procedure **usp_inout** taking one INOUT-type int argument 'p' is defined. Inside this procedure, p is incremented by 2. To call procedure having INOUT type argument, it is compulsory to provide variable name instead of value otherwise error will be generated. So, global variable @param_1 is created, assigned value, and provided as input argument during function call and result is displayed.

Procedure:


```

01. delimiter $$
02. create procedure GetAllProducts()
03. begin
04. select * from products;
05. end$$
06. delimiter ;

```

Detail: Any valid SQL query can be specified inside the procedure. This example shows a procedure that retrieves and displays all the products from product table using simple select statement.

Procedure:



```
01. delimiter $$
02. create procedure getdata()
03. begin
04. declare a varchar(20);
05. declare b tinytext;
06. select user_name, user_password into a,b from user_data limit 1;
07. select a,b;
08. end$$
09. delimiter ;
```

Detail: This procedure demonstrates three things: 1) how local variable can be declared, 2) how data is retrieved and stored in locally defined variable, and 3) how to display content of a local variable.

Procedure:

```
01. create table tmptest(id int, txt varchar(10), primary key(id));
02. select * from tmptest;
03.
04. delimiter $$
05. create procedure usp_variable(in p int)
06. begin
07. declare a int;
08. declare b int default 10;
09. set a = p*b;
10. insert into tmptest(id, txt) values (a,hex(DEF));
11. end$$
12. delimiter ;
13.
14. call usp_variable(4);
15. select * from tmptest;
```

Detail: This example defines a procedure that takes data from user and stores it in its respective table. A new table **tmptest** is created and its content is displayed. Next, procedure is called and provided value on run-time. Then, contents of updated table are displayed.
