Stored Procedures

What is a Stored Procedure?

A **stored procedure** or in simple a **proc** is a named PL/SQL block which performs one or more specific task. This is similar to a procedure in other programming languages.

A procedure has a header and a body. The header consists of the name of the procedure and the parameters or variables passed to the procedure. The body consists or declaration section, execution section and exception section similar to a general PL/SQL Block.

A procedure is similar to an anonymous PL/SQL Block but it is named for repeated usage.

**Procedures: Passing Parameters**

We can pass parameters to procedures in three ways.  
1) IN-parameters  
2) OUT-parameters  
3) IN OUT-parameters

A procedure may or may not return any value.

**General Syntax to create a procedure is:**

*CREATE [OR REPLACE] PROCEDURE proc\_name [list of parameters]*

*IS*

*Declaration section*

*BEGIN*

*Execution section*

*EXCEPTION*

*Exception section*

*END;*

**IS -** marks the beginning of the body of the procedure and is similar to DECLARE in anonymous PL/SQL Blocks. The code between IS and BEGIN forms the Declaration section.

The syntax within the brackets [ ] indicate they are optional. By using CREATE OR REPLACE together the procedure is created if no other procedure with the same name exists or the existing procedure is replaced with the current code.

**Procedures: Example**

The below example creates a procedure ‘employer\_details’ which gives the details of the employee.

*1> CREATE OR REPLACE PROCEDURE employer\_details*

*2> IS*

*3> CURSOR emp\_cur IS*

*4> SELECT first\_name, last\_name, salary FROM emp\_tbl;*

*5> emp\_rec emp\_cur%rowtype;*

*6> BEGIN*

*7> FOR emp\_rec in sales\_cur*

*8> LOOP*

*9> dbms\_output.put\_line(emp\_cur.first\_name || ' ' ||emp\_cur.last\_name*

*10> || ' ' ||emp\_cur.salary);*

*11> END LOOP;*

*12>END;*

*13> /*

How to execute a Stored Procedure?

There are two ways to execute a procedure.

1) From the SQL prompt.

*EXECUTE [or EXEC] procedure\_name;*

2) Within another procedure – simply use the procedure name.

*procedure\_name;*

**NOTE:** In the examples given above, we are using backward slash ‘/’ at the end of the program. This indicates the oracle engine that the PL/SQL program has ended and it can begin processing the statements.

|  |  |
| --- | --- |
| PL/SQL Functions  What is a Function in PL/SQL?  A function is a named PL/SQL Block which is similar to a procedure. The major difference between a procedure and a function is, a function must always return a value, but a procedure may or may not return a value.  **General Syntax to create a function is**  *CREATE [OR REPLACE] FUNCTION function\_name [parameters]*  *RETURN return\_datatype;*  *IS*  *Declaration\_section*  *BEGIN*  *Execution\_section*  *Return return\_variable;*  *EXCEPTION*  *exception section*  *Return return\_variable;*  *END;* |  |

1) **Return Type:** The header section defines the return type of the function. The return datatype can be any of the oracle datatype like varchar, number etc.  
2) The execution and exception section both should return a value which is of the datatype defined in the header section.

For example, let’s create a frunction called ''employer\_details\_func' similar to the one created in stored proc

*1> CREATE OR REPLACE FUNCTION employer\_details\_func*

*2> RETURN VARCHAR(20);*

*3> IS*

*5> emp\_name VARCHAR(20);*

*6> BEGIN*

*7> SELECT first\_name INTO emp\_name*

*8> FROM emp\_tbl WHERE empID = '100';*

*9> RETURN emp\_name;*

*10> END;*

*11> /*

In the example we are retrieving the ‘first\_name’ of employee with empID 100 to variable ‘emp\_name’.  
The return type of the function is VARCHAR which is declared in line no 2.   
The function returns the 'emp\_name' which is of type VARCHAR as the return value in line no 9.

How to execute a PL/SQL Function?

A function can be executed in the following ways.

1) Since a function returns a value we can assign it to a variable.

*employee\_name := employer\_details\_func;*

If ‘employee\_name’ is of datatype varchar we can store the name of the employee by assigning the return type of the function to it.

2) As a part of a SELECT statement

*SELECT employer\_details\_func FROM dual;*

3) In a PL/SQL Statements like,

*dbms\_output.put\_line(employer\_details\_func);*

This line displays the value returned by the function.

Parameters in Procedure and Functions

How to pass parameters to Procedures and Functions in PL/SQL?

In PL/SQL, we can pass parameters to procedures and functions in three ways.

**1) IN type parameter:** These types of parameters are used to send values to stored procedures.   
**2) OUT type parameter:** These types of parameters are used to get values from stored procedures. This is similar to a return type in functions.  
**3) IN OUT parameter:**These types of parameters are used to send values and get values from stored procedures.

**NOTE:**If a parameter is not explicitly defined a parameter type, then by default it is an IN type parameter.

|  |  |
| --- | --- |
| **1) IN parameter:**  This is similar to passing parameters in programming languages. We can pass values to the stored procedure through these parameters or variables. This type of parameter is a read only parameter. We can assign the value of IN type parameter to a variable or use it in a query, but we cannot change its value inside the procedure. General syntax to pass a IN parameter is |  |

*CREATE [OR REPLACE] PROCEDURE procedure\_name (*

*param\_name1 IN datatype, param\_name12 IN datatype ... )*

* param\_name1, param\_name2... are unique parameter names.
* datatype - defines the datatype of the variable.
* IN - is optional, by default it is a IN type parameter.

**2) OUT Parameter:**

The OUT parameters are used to send the OUTPUT from a procedure or a function. This is a write-only parameter i.e, we cannot pass values to OUT paramters while executing the stored procedure, but we can assign values to OUT parameter inside the stored procedure and the calling program can recieve this output value.

The General syntax to create an OUT parameter is

*CREATE [OR REPLACE] PROCEDURE proc2 (param\_name OUT datatype)*

The parameter should be explicity declared as OUT parameter.   
  
**3) IN OUT Parameter:**

The IN OUT parameter allows us to pass values into a procedure and get output values from the procedure. This parameter is used if the value of the IN parameter can be changed in the calling program.

By using IN OUT parameter we can pass values into a parameter and return a value to the calling program using the same parameter. But this is possible only if the value passed to the procedure and output value have a same datatype. This parameter is used if the value of the parameter will be changed in the procedure.

The General syntax to create an IN OUT parameter is

*CREATE [OR REPLACE] PROCEDURE proc3 (param\_name IN OUT datatype)*

The below examples show how to create stored procedures using the above three types of parameters.

Example1:

**Using IN and OUT parameter:**

Let’s create a procedure which gets the name of the employee when the employee id is passed.

*1> CREATE OR REPLACE PROCEDURE emp\_name (id IN NUMBER, emp\_name OUT NUMBER)*

*2> IS*

*3> BEGIN*

*4> SELECT first\_name INTO emp\_name*

*5> FROM emp\_tbl WHERE empID = id;*

*6> END;*

*7> /*

We can call the procedure ‘emp\_name’ in this way from a PL/SQL Block.

*1> DECLARE*

*2> empName varchar(20);*

*3> CURSOR id\_cur SELECT id FROM emp\_ids;*

*4> BEGIN*

*5> FOR emp\_rec in id\_cur*

*6> LOOP*

*7> emp\_name(emp\_rec.id, empName);*

*8> dbms\_output.putline('The employee ' || empName || ' has id ' || emp-rec.id);*

*9> END LOOP;*

*10> END;*

*11> /*

In the above PL/SQL Block   
In line no 3; we are creating a cursor ‘id\_cur’ which contains the employee id.  
In line no 7; we are calling the procedure ‘emp\_name’, we are passing the ‘id’ as IN parameter and ‘empName’ as OUT parameter.  
In line no 8; we are displaying the id and the employee name which we got from the procedure ‘emp\_name’.

Example 2:

**Using IN OUT parameter in procedures:**

*1> CREATE OR REPLACE PROCEDURE emp\_salary\_increase*

*2> (emp\_id IN emptbl.empID%type, salary\_inc IN OUT emptbl.salary%type)*

*3> IS*

*4> tmp\_sal number;*

*5> BEGIN*

*6> SELECT salary*

*7> INTO tmp\_sal*

*8> FROM emp\_tbl*

*9> WHERE empID = emp\_id;*

*10> IF tmp\_sal between 10000 and 20000 THEN*

*11> salary\_inout := tmp\_sal \* 1.2;*

*12> ELSIF tmp\_sal between 20000 and 30000 THEN*

*13> salary\_inout := tmp\_sal \* 1.3;*

*14> ELSIF tmp\_sal > 30000 THEN*

*15> salary\_inout := tmp\_sal \* 1.4;*

*16> END IF;*

*17> END;*

*18> /*

The below PL/SQL block shows how to execute the above 'emp\_salary\_increase' procedure.

*1> DECLARE*

*2> CURSOR updated\_sal is*

*3> SELECT empID,salary*

*4> FROM emp\_tbl;*

*5> pre\_sal number;*

*6> BEGIN*

*7> FOR emp\_rec IN updated\_sal LOOP*

*8> pre\_sal := emp\_rec.salary;*

*9> emp\_salary\_increase(emp\_rec.empID, emp\_rec.salary);*

*10> dbms\_output.put\_line('The salary of ' || emp\_rec.empID ||*

*11> ' increased from '|| pre\_sal || ' to '||emp\_rec.salary);*

*12> END LOOP;*

*13> END;*

*14> /*