**Procedure in PL/SQL?**

A Procedure is a subprogram unit that consists of a group of PL/SQL statements. Each procedure in Oracle has its own unique name by which it can be referred. This subprogram unit is stored as a database object.

**Characteristics of Procedure:**

* Procedures are standalone blocks of a program that can be stored in the database.
* Call to these procedures can be made by referring to their name, to execute the PL/SQL statements.
* It is mainly used to execute a process in PL/SQL.
* It can have nested blocks, or it can be defined and nested inside the other blocks or packages.
* It contains declaration part (optional), execution part, exception handling part (optional).
* The values can be passed into the procedure or fetched from the procedure through parameters.
* These parameters should be included in the calling statement.
* Procedure can have a RETURN statement to return the control to the calling block, but it cannot return any values through the RETURN statement.
* Procedures cannot be called directly from SELECT statements. They can be called from another block or through EXEC keyword.

### Parameter:

The parameter is variable or placeholder of any valid PL/SQL datatype through which the PL/SQL subprogram exchange the values with the main code. This parameter allows to give input to the subprograms and to extract from these subprograms.

* These parameters should be defined along with the subprograms at the time of creation.
* These parameters are included n the calling statement of these subprograms to interact the values with the subprograms.
* The datatype of the parameter in the subprogram and the calling statement should be same.
* The size of the datatype should not mention at the time of parameter declaration, as the size is dynamic for this type.

Based on their purpose parameters are classified as

1. IN Parameter
2. OUT Parameter
3. IN OUT Parameter

### IN Parameter:

* This parameter is used for giving input to the subprograms.
* It is a read-only variable inside the subprograms. Their values cannot be changed inside the subprogram.
* In the calling statement, these parameters can be a variable or a literal value or an expression, for example, it could be the arithmetic expression like '5\*8' or 'a/b' where 'a' and 'b' are variables.
* By default, the parameters are of IN type.

### OUT Parameter:

* This parameter is used for getting output from the subprograms.
* It is a read-write variable inside the subprograms. Their values can be changed inside the subprograms.
* In the calling statement, these parameters should always be a variable to hold the value from the current subprograms.

### IN OUT Parameter:

* This parameter is used for both giving input and for getting output from the subprograms.
* It is a read-write variable inside the subprograms. Their values can be changed inside the subprograms.
* In the calling statement, these parameters should always be a variable to hold the value from the subprograms.

#### Syntax:

CREATE OR REPLACE PROCEDURE

<procedure\_name>

(

<parameterl IN/OUT <datatype>

..

.

)

[ IS | AS ]

<declaration\_part>

BEGIN

<execution part>

EXCEPTION

<exception handling part>

END;

* CREATE PROCEDURE instructs the compiler to create new procedure. Keyword 'OR REPLACE' instructs the compile to replace the existing procedure (if any) with the current one.
* Procedure name should be unique.
* Keyword 'IS' will be used, when the procedure is nested into some other blocks. If the procedure is standalone then 'AS' will be used. Other than this coding standard, both have the same meaning.

**Example1: Creating Procedure and calling it using EXEC**

In this example, we are going to create a procedure that takes the name as input and prints the welcome message as output. We are going to use EXEC command to call procedure.

CREATE OR REPLACE PROCEDURE welcome\_msg (p\_name IN VARCHAR2)

IS

BEGIN

dbms\_output.put\_line (‘Welcome '|| p\_name);

END;

/

EXEC welcome\_msg (‘Guru99’);

## Function?

Functions is a standalone PL/SQL subprogram. Like PL/SQL procedure, functions have a unique name by which it can be referred. These are stored as PL/SQL database objects. Below are some of the characteristics of functions.

* Functions are a standalone block that is mainly used for calculation purpose.
* Function use RETURN keyword to return the value, and the datatype of this is defined at the time of creation.
* A Function should either return a value or raise the exception, i.e. return is mandatory in functions.
* Function with no DML statements can be directly called in SELECT query whereas the function with DML operation can only be called from other PL/SQL blocks.
* It can have nested blocks, or it can be defined and nested inside the other blocks or packages.
* It contains declaration part (optional), execution part, exception handling part (optional).
* The values can be passed into the function or fetched from the procedure through the parameters.
* These parameters should be included in the calling statement.
* Function can also return the value through OUT parameters other than using RETURN.
* Since it will always return the value, in calling statement it always accompanies with assignment operator to populate the variables.

#### Syntax

CREATE OR REPLACE FUNCTION

<procedure\_name>

(

<parameterl IN/OUT <datatype>

)

RETURN <datatype>

[ IS | AS ]

<declaration\_part>

BEGIN

<execution part>

EXCEPTION

<exception handling part>

END;

* CREATE FUNCTION instructs the compiler to create a new function. Keyword 'OR REPLACE' instructs the compiler to replace the existing function (if any) with the current one.
* The Function name should be unique.
* RETURN datatype should be mentioned.
* Keyword 'IS' will be used, when the procedure is nested into some other blocks. If the procedure is standalone then 'AS' will be used. Other than this coding standard, both have the same meaning.

**Example1: Creating Function and calling it using Anonymous Block**

In this program, we are going to create a function that takes the name as input and returns the welcome message as output. We are going to use anonymous block and select statement to call the function.

CREATE OR REPLACE FUNCTION welcome\_msgJune ( p\_name IN VARCHAR2) RETURN VAR.CHAR2

IS

BEGIN

RETURN (‘Welcome ‘|| p\_name);

END;

/

DECLARE

lv\_msg VARCHAR2(250);

BEGIN

lv\_msg := welcome\_msg\_func (‘Guru99’);

dbms\_output.put\_line(lv\_msg);

END;

SELECT welcome\_msg\_func(‘Guru99:) FROM DUAL;

**Similarities between Procedure and Function**

* Both can be called from other PL/SQL blocks.
* If the exception raised in the subprogram is not handled in the subprogram exception handling section, then it will propagate to the calling block.
* Both can have as many parameters as required.
* Both are treated as database objects in PL/SQL.

**Procedure vs. Function: Key Differences**

| **Procedure** | **Function** |
| --- | --- |
| * Used mainly to a execute certain process | * Used mainly to perform some calculation |
| * Cannot call in SELECT statement | * A Function that contains no DML statements can be called in SELECT statement |
| * Use OUT parameter to return the value | * Use RETURN to return the value |
| * It is not mandatory to return the value | * It is mandatory to return the value |
| * RETURN will simply exit the control from subprogram. | * RETURN will exit the control from subprogram and also returns the value |
| * Return datatype will not be specified at the time of creation | * Return datatype is mandatory at the time of creation |

Stored procedure for SYSDATE:

CREATE PROCEDURE getCurrentDate

IS

datevalue DATE;

BEGIN

SELECT SYSDATE

INTO datevalue

FROM dual;

END;

**SELECT STORED PROCEDURE:**

CREATE PROCEDURE Employees\_Details

IS

BEGIN

SELECT first\_name, last\_name, date\_of\_birth, address

FROM employees;

END;

**SQL view:**

SQL view is nothing but a logical table or a virtual table stored in a database. We can also define a VIEW as SELECT Statement with a name which is stored in a database as though it were a table. All the DML commands which you can perform on a table can be performed on a view also.

### **Use of SQL View in Oracle Database**

Views in any database are majorly used for two reasons –

1. Security
2. Reducing complexity of a query.

#### Security:

Using a VIEW you can mask columns of a table and restrict any user to use the data from that column. For example,

Suppose you have a large table containing a mixture of both sensitive as well as general interest information. In this case it will be very handy for you to create a view which only queries the general interest columns of the original table and in turn grants privileges on this view to the general users. In this case the population of general users can query the view and have direct access only to the general information omitting the sensitive information present in the underlying table. Thus in this way views can be used to give access to selective data in a table.

#### Reducing the complexity of a query

Another reason for using a VIEW is to reduce the complexity of a query which is made by joining various tables. For example,

A view built on a complex join can be created in order to include the complexity in the view itself. The result of this is a regular view object that looks to be a single table which could be queried by you as any regular table. The view can be joined with other views and tables as well. Thus here in this way, view can be used to reduce the complexity of a common Join.

### **Syntax**

The syntax for the CREATE OR REPLACE VIEW Statement in Oracle/PLSQL is:

CREATE OR REPLACE VIEW view\_name AS

SELECT columns

FROM table

WHERE conditions;

#### **Create a Simple View**

In this example I will create a view by the name of vw\_rebellion on Employees table of HR schema. You can give whatever name you want to your view.

**CREATE VIEW  Emp\_View  AS   
  SELECT  first\_name,  email,  phone\_number  FROM  employees;**

#### **How To retrieve Rows from a VIEW in Oracle Database:**

You can use SELECT ddl command to retrieve rows from a VIEW similar to the way we do with tables in oracle database.

**SELECT \* FROM Emp\_View**

#### **How to create a complex view (View by joining two tables)**

Complexity of a view depends upon the complexity of its SELECT statement. You can increase the complexity of a view by increasing the complexity of the SELECT statement.

**CREATE  OR  REPLACE  VIEW  vw\_join  AS  
  
  SELECT  first\_name,  department\_name  FROM  employees emp  
  
   FULL  OUTER  JOIN  departments dept  
  
  ON  (emp.DEPARTMENT\_ID = dept.DEPARTMENT\_ID);**