**Overview of Stored Procedures**

You already know how to interact with the database using SQL, but it is not sufficient for building enterprise applications. PL/SQL is a third generation language that has the expected procedural and namespace constructs, and its tight integration with SQL makes it possible to build complex and powerful applications. Because PL/SQL is executed in the database, you can include SQL statements in your code without having to establish a separate connection.

The main types of program units you can create with PL/SQL and store in the database are standalone procedures and functions, and packages. Once stored in the database, these PL/SQL components, collectively known as **stored procedures**, can be used as building blocks for several different applications.

While standalone procedures and functions are invaluable for testing pieces of program logic, Oracle recommends that you place all your code inside a package. Packages are easier to port to another system and have the additional benefit of qualifying the names of your program units with the package name. For example, if you developed a schema-level procedure called continue in a previous version of Oracle Database, your code would not compile when you port it to a newer Oracle Database installation. This is because Oracle recently introduced the statement CONTINUE that exits the current iteration of a loop and transfers control to the next iteration. If you developed your procedure inside a package, the procedure *package\_name*.continue would have been protected from such name capture.

This next section of this chapter is ["Creating and Using Standalone Procedures and Functions"](https://docs.oracle.com/cd/B28359_01/appdev.111/b28843/tdddg_procedures.htm#CIHCHIDB), shows you how to create and use standalone procedures and functions. You may wish to skip it and move directly to ["Creating and Using Packages"](https://docs.oracle.com/cd/B28359_01/appdev.111/b28843/tdddg_procedures.htm#CIHGAGJG).

Creating and Using Standalone Procedures and Functions

With Oracle Database, you can store programs in the database, so commonly used code can be written and tested once and then accessed by any application that requires it. Program units that reside in the database also ensure that when the code is invoked the data is processed consistently, which leads to ease and consistency of the application development process.

Schema-level, or standalone subprograms such as functions (which return a value) and procedures (which do not return a value) are compiled and stored in an Oracle Database. Once compiled, they become **stored procedure** or **stored function** schema objects, and can be referenced or called by any applications connected to Oracle Database. At invocation, both stored procedures and functions can accept parameters.

Procedures and functions follow the basic PL/SQL block structure, which consists of the following elements:

* A declarative part, sometimes starting with the keyword DECLARE, identifies variables and constants used in the application logic. This part is optional.
* An executable part, starting with BEGIN and ending with END, contains the application logic. This part is mandatory.
* An exception-handling part, starting with EXCEPTION, handles error conditions that may be raised in the executable part of the block. This part is optional.

The general form of a PL/SQL block follows. Note also that each stored program unit has a header that names the unit and identifies it as either a function, procedure, or a package.

*Header* AS

[declaration statements

...]

BEGIN

...

[EXCEPTION

...]

END;

**Creating Procedures and Functions**

The SQL statements for creating procedures and functions are CREATE PROCEDURE and CREATE FUNCTION, respectively. In practice, it is best to use a CREATE OR REPLACE statement. The general form of these statements follows.

CREATE OR REPLACE *procedure\_name*(*arg1* *data\_type,* ...) AS

BEGIN

....

END *procedure\_name*;

CREATE OR REPLACE *procedure\_name*(*arg1* *data\_type,* ...) AS

BEGIN

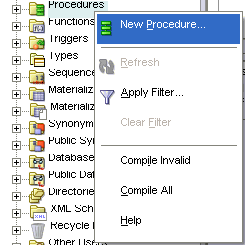
....

END *procedure\_name*;

**To create a procedure:**

You will create a procedure add\_evaluation that creates a new row in the evaluations table.

1. In the Connections navigation hierarchy, right-click **Procedures**.
2. Select **New Procedure**.

  
[Description of the illustration create\_procedure\_1.gif](https://docs.oracle.com/cd/B28359_01/appdev.111/b28843/img_text/create_procedure_1.htm)

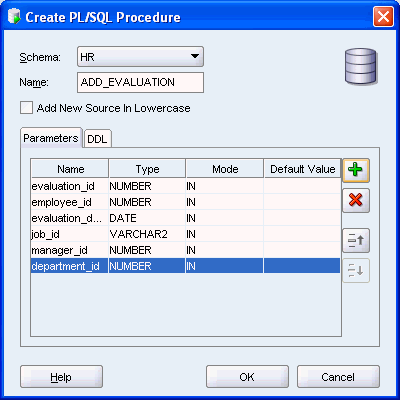
1. In the New Procedure window, set the following parameters:
   * Ensure that **Schema** is set to HR.
   * Set **Name** to ADD\_EVALUATION.

In the Parameters tab, click the **Add Column** icon ('plus' sign) and specify the first parameter of the procedure. Set **Name** to eval\_id, set **Type** to NUMBER, set **Mode** to IN, and leave **Default Value** empty.

Similarly, add the following parameters, in this order:

* + employee\_id: set **Type** to NUMBER, set **Mode** to IN, and leave **Default Value** empty.
  + evaluation\_date: set **Type** to DATE, set **Mode** to IN, and leave **Default Value** empty.
  + job\_id: set **Type** to VARCHAR2, set **Mode** to IN, and leave **Default Value** empty.
  + manager\_id: set **Type** to NUMBER, set **Mode** to IN, and leave **Default Value** empty
  + department\_id: set **Type** to NUMBER, set **Mode** to IN, and leave **Default Value** empty

Click **OK**.

  
[Description of the illustration create\_procedure\_2.gif](https://docs.oracle.com/cd/B28359_01/appdev.111/b28843/img_text/create_procedure_2.htm)

1. The ADD\_EVALUATION pane opens with the following code.

Note that the tile of the pane is in italic font, which indicates that the procedure is not saved in the database.

CREATE OR REPLACE

PROCEDURE ADD\_EVALUATION

( evaluation\_id IN NUMBER

, employee\_id IN NUMBER

, evaluation\_date IN DATE

, job\_id IN VARCHAR2

, manager\_id IN NUMBER

, department\_id IN NUMBER

) AS

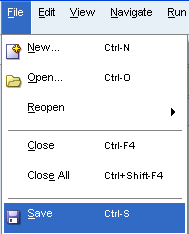
BEGIN

NULL;

END ADD\_EVALUATION;

1. From the **File** menu, select **Save** to save the new procedures. Alternatively, use the **CTRL + S** key combination.

Note that Oracle Database automatically compiles procedures prior to saving them.

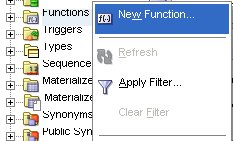
  
[Description of the illustration create\_procedure\_3.gif](https://docs.oracle.com/cd/B28359_01/appdev.111/b28843/img_text/create_procedure_3.htm)

Note that the tile of the add\_evaluation pane is in regular font, not italic; this indicates that the procedure is saved to the database

**To create a function:**

You will create a new function calculate\_score, which calculates the weighted score based on the performance in a particular category.

1. In the Connections navigation hierarchy, right-click **Functions**.
2. Select **New Function**.

  
[Description of the illustration create\_function\_1.gif](https://docs.oracle.com/cd/B28359_01/appdev.111/b28843/img_text/create_function_1.htm)

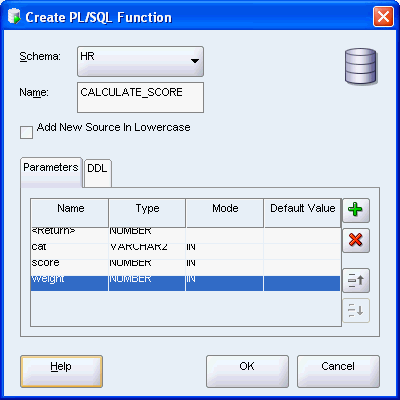
1. In the New Function window, set the following parameters:
   * Ensure that **Schema** is set to HR.
   * Set **Name** to CALCULATE\_SCORE.

In the Parameters pane, set the **<return>** **Type** to NUMBER.

Similarly, add the following parameters, in this order:

* + cat: set **Type** to VARCHAR2, set **Mode** to IN, and leave **Default Value** empty.
  + score: set **Type** to NUMBER, set **Mode** to IN, and leave **Default Value** empty
  + weight: set **Type** to NUMBER, set **Mode** to IN, and leave **Default Value** empty

Click **OK**.

  
[Description of the illustration create\_function\_2.gif](https://docs.oracle.com/cd/B28359_01/appdev.111/b28843/img_text/create_function_2.htm)

1. The calculate\_score pane opens with the following code.

Note that the tile of the pane is in italic font, which indicates that the procedure is not saved in the database.

CREATE OR REPLACE

FUNCTION calculate\_score

( cat IN VARCHAR2

, score IN NUMBER

, weight IN NUMBER

) RETURN NUMBER AS

BEGIN

RETURN NULL;

END calculate\_score;

1. From the **File** menu, select **Save** to save the new function. Alternatively, use the **CTRL + S** key combination.

Note that Oracle Database automatically compiles functions prior to saving them.

Note that the tile of the calculate\_score pane is in regular font, not italic; this indicates that the procedure is saved to the database

**Modifying Procedures and Functions**

You already created a new procedure and a new function. However, they both consist of only the subprogram signature. In this section, you will edit a subprogram body.

**To modify a function:**

You will edit the function calculate\_score to determine the weighted value of an evaluation for a particular category.

1. In the calculate\_score pane, replace the body of the function with the following code. The new code is in bold font.
2. BEGIN
3. RETURN **score \* weight**;
4. END calculate\_score;
5. Compile and save the function; you may use the **CTRL + S** key combination.

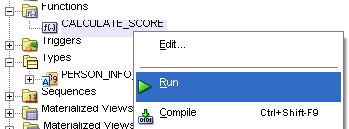
**Testing Procedures and Functions**

Next, you will test the function that you just modified.

**To test a function:**

You will test the function calculate\_score.

1. In the Connections navigator hierarchy, right-click the calculate\_score function. Select **Run**.

  
[Description of the illustration run\_function\_1.gif](https://docs.oracle.com/cd/B28359_01/appdev.111/b28843/img_text/run_function_1.htm)

1. In the Run PL/SQL window, click inside the PL/SQL Block pane, and edit the assignments for the score and weight variables. The new code is in bold font.
2. v\_Return := CALCULATE\_SCORE(
3. CAT => CAT,
4. SCORE => **8**,
5. WEIGHT => **0.2**
6. );

Click **OK**.

1. In the Running - Log pane, note the following results:
2. Connecting to the database hr\_conn.
3. v\_Return = 1.6
4. Process exited.
5. Disconnecting from the database hr\_conn.

**Dropping Procedures and Functions**

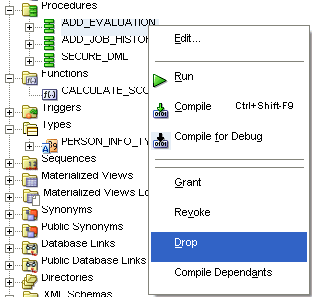
You can delete a procedure or function from the database using either the Connection Navigator, or the SQL DROP statement.

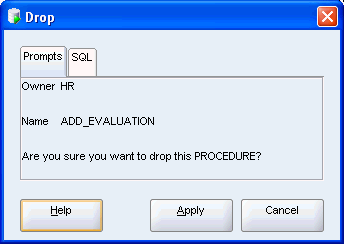
**To drop a procedure:**

You will drop the procedure ADD\_EVALUATION.

1. In the Connections navigator hierarchy, right-click the ADD\_EVALUATION function.

Select **Drop**.

  
[Description of the illustration drop\_procedure\_1.gif](https://docs.oracle.com/cd/B28359_01/appdev.111/b28843/img_text/drop_procedure_1.htm)

1. In the Drop window, click **Apply**.
2.   
   [Description of the illustration drop\_procedure\_2.gif](https://docs.oracle.com/cd/B28359_01/appdev.111/b28843/img_text/drop_procedure_2.htm)
3. In the Confirmation dialog box, click **OK**.

You dropped the ADD\_EVALUATION procedure from the database