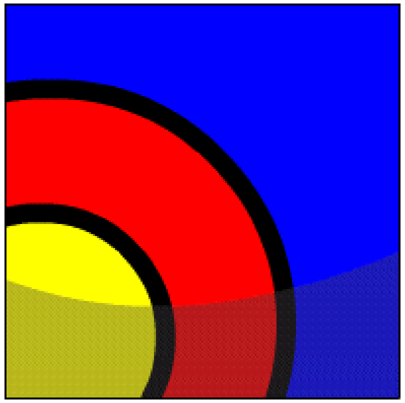
Creating Functions

What Will I Learn?

In this lesson, you will learn to:

- Define a stored function
- Create a PL/SQL block containing a function
- List ways in which you can invoke a function
- Create a PL/SQL block that invokes a function that has parameters
- List the development steps for creating a function
- Describe the differences between procedures and functions



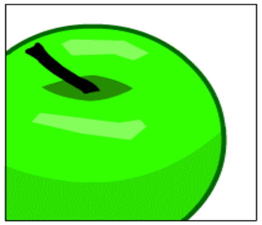
Why Learn It?

In this lesson, you learn how to create and invoke functions. A function is a subprogram that must return exactly one value.

A procedure is a standalone executable statement, whereas a function can only exist as part of an executable statement.

Functions are an integral part of modular code. Business rules and/or formulas can be placed in functions so that they can be easily reused.

Tell Me / Show Me



What Is a Stored Function?

- A function is a named PL/SQL block (a subprogram) that can accept optional IN parameters and must return a single output value.

- Functions are stored in the database as schema objects for repeated execution.

Tell Me / Show Me

What Is a Stored Function? (continued)

- A function can be called as part of an SQL expression or as part of a PL/SQL expression.
  - Certain return types, for example, Boolean, prevent a function from being called as part of a `SELECT`.
- In SQL expressions, a function must obey specific rules to control side effects. Side effects to be avoided are:
  - Any kind of DML or DDL
  - `COMMIT` or `ROLLBACK`
  - Altering global variables
- In PL/SQL expressions, the function identifier acts like a variable whose value depends on the parameters passed to it.

Tell Me / Show Me

Syntax for Creating Functions

The PL/SQL block must have at least one RETURN statement.

```
CREATE [OR REPLACE] FUNCTION function_name
 [(parameter1 [mode1] datatype1, ...)]
RETURN datatype IS|AS
 [local_variable_declarations; …]
BEGIN
   -- actions;
   RETURN expression;
END [function_name];
```

PL/SQL Block

The header is like a PROCEDURE header with two differences:

1. The parameter mode should only be IN.
2. The RETURN clause is used instead of an OUT mode.

Tell Me / Show Me

Syntax for Creating Functions (continued)

- A function is a PL/SQL subprogram that returns a single value. You must provide a `RETURN` statement to return a value with a data type that is consistent with the function declaration type.

- You create new functions with the `CREATE [OR REPLACE] FUNCTION` statement, which can declare a list of parameters, must return exactly one value, and must define the actions to be performed by the PL/SQL block.

Tell Me / Show Me

Stored Function With a Parameter: Example

- Create the function:

```
CREATE OR REPLACE FUNCTION get_sal
 (p_id employees.employee_id%TYPE)
  RETURN NUMBER IS
  v_sal employees.salary%TYPE := 0;
BEGIN
  SELECT salary
    INTO  v_sal
    FROM  employees
    WHERE employee_id = p_id;
  RETURN v_sal;
END get_sal;
```

- Invoke the function as an expression or as a parameter value:

```
 ... v_salary := get_sal(100);
```

Tell Me / Show Me

You can RETURN from the executable section and/or from the EXCEPTION section.

- Create the function

```
CREATE OR REPLACE FUNCTION get_sal
 (p_id employees.employee_id%TYPE) RETURN NUMBER IS
  v_sal employees.salary%TYPE := 0;
BEGIN
  SELECT salary INTO v_sal
    FROM employees WHERE employee_id = p_id;
  RETURN v_sal;
EXCEPTION
  WHEN NO_DATA_FOUND THEN RETURN NULL;
END get_sal;
```

- Invoke the function as an expression with a bad parameter

```
... v_salary := get_sal(999);
```

Tell Me / Show Me

Ways to Invoke (or Execute) Functions With Parameters

- Invoke as part of a PL/SQL expression, using a local variable to store the returned result

```
DECLARE v_sal employees.salary%type;
BEGIN
  v_sal := get_sal(100); ...
END;
```

A

- Use as a parameter to another subprogram

```
... DBMS_OUTPUT.PUT_LINE(get_sal(100));
```

B

- Use in an SQL statement (subject to restrictions)

```
SELECT job_id, get_sal(employee_id) FROM employees;
```

C

Tell Me / Show Me

Ways to Invoke (or Execute) Functions With Parameters

If functions are designed thoughtfully, they can be powerful constructs. You can invoke functions in the following ways:

- As part of PL/SQL expressions: (A) Uses a local variable in an anonymous block to hold the returned value from a function.
- As a parameter to another subprogram: (B) Demonstrates this usage. The `get_sal` function with all its arguments is nested in the parameter required by the `DBMS_OUTPUT.PUT_LINE` procedure.
- As an expression in an SQL statement: (C) Shows how you can use a function as a single-row function in an SQL statement.

Note: The restrictions that apply to functions when used in an SQL statement are discussed in the next lesson.

Tell Me / Show Me

Invoking Functions Without Parameters

Most functions have parameters, but not all. The following are system functions USER and SYSDATE without parameters.

- Invoke as part of a PL/SQL expression, using a local variable to obtain the result

```
DECLARE v_today DATE;
BEGIN
  v_today := SYSDATE; ...
END;
```

- Use as a parameter to another subprogram

```
... DBMS_OUTPUT.PUT_LINE(USER);
```

- Use in an SQL statement (subject to restrictions)

```
SELECT job_id, SYSDATE-hiredate FROM employees;
```

Tell Me / Show Me

Benefits and Restrictions That Apply to Functions

+ Try things quickly: Functions allow you to temporarily display a value in a new format: a different case, annually vs. monthly (times 12), concatenated or with substrings.
+ Extend functionality: Add new features, such as spell checking and parsing.
– Restrictions: PL/SQL types do not completely overlap with SQL types. What is fine for PL/SQL (for example, `BOOLEAN`, `RECORD`) might be invalid for a `SELECT`.
– Restrictions: PL/SQL sizes are not the same as SQL sizes. For instance, a PL/SQL VARCHAR2 variable can be up to 32 KB, whereas an SQL VARCHAR2 column can be only up to 4 KB.

Tell Me / Show Me

Syntax Differences Between Procedures and Functions

Procedures

```
CREATE [OR REPLACE] PROCEDURE name [parameters] IS|AS (Mandatory)
  Variables, cursors, etc. (Optional)
BEGIN               (Mandatory)
  SQL and PL/SQL statements;
EXCEPTION           (Optional)
  WHEN exception-handling actions;
END [name];           (Mandatory)
```

Functions

```
CREATE [OR REPLACE] FUNCTION name [parameters] (Mandatory)
  RETURN datatype IS|AS          (Mandatory)
  Variables, cursors, etc. (Optional)
BEGIN               (Mandatory)
  SQL and PL/SQL statements;
  RETURN ...; (One Mandatory, more optional)
EXCEPTION           (Optional)
  WHEN exception-handling actions;
END [name];           (Mandatory)
```

Tell Me / Show Me

Differences/Similarities Between Procedures and Functions

Procedures	Functions
Execute as a PL/SQL statement	Invoke as part of an expression
Do not contain RETURN clause in the header	Must contain a RETURN clause in the header
Can return values (if any) in output parameters	Must return a single value
Can contain a RETURN statement without a value	Must contain at least one RETURN statement

Both can have zero or more IN parameters that can be passed from the calling environment.
Both have the standard block structure including exception handling.

Tell Me / Show Me

Differences Between Procedures and Functions

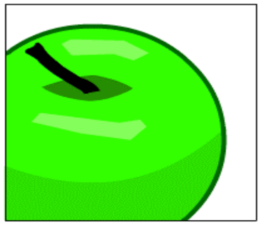
Procedures

- You create a procedure to store a series of actions for later execution. A procedure does not have to return a value. A procedure can call a function to assist with its actions. **Note:** A procedure containing a single `OUT` parameter might be better rewritten as a function returning the value.

Functions

- You create a function when you want to compute a value that must be returned to the calling environment. Functions return only a single value, and the value is returned through a `RETURN` statement. The functions used in SQL statements cannot use `OUT` or `IN OUT` modes. Although a function using `OUT` can be invoked from a PL/SQL procedure or anonymous block, it cannot be used in SQL statements.

Tell Me / Show Me



Terminology

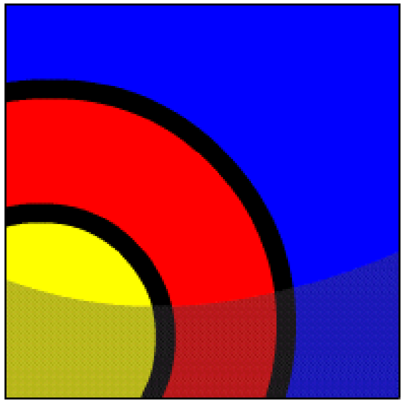
Key terms used in this lesson include:

Stored function

Summary

In this lesson, you learned to:

- Define a stored function
- Create a PL/SQL block containing a function
- List ways in which a function can be invoked
- Create a PL/SQL block that invokes a function that has parameters
- List the development steps for creating a function
- Describe the differences between procedures and functions



Try It / Solve It

The exercises in this lesson cover the following topics:

- Defining a stored function
- Creating a function
- Listing how a function can be invoked
- Invoking a function that has parameters
- Listing the development steps for creating a function
- Describing the differences between procedures and functions