# 10

# Creating Functions

Copyright © Oracle Corporation, 2001. All rights reserved.

# Objectives

After completing this lesson, you should be able to do the following:

- Describe the uses of functions
- Create stored functions
- Invoke a function
- Remove a function
- Differentiate between a procedure and a function


Copyright © Oracle Corporation, 2001. All rights reserved.

# Overview of Stored Functions

- A function is a named PL/SQL block that returns a value.
- A function can be stored in the database as a schema object for repeated execution.
- A function is called as part of an expression.


Copyright © Oracle Corporation, 2001. All rights reserved.

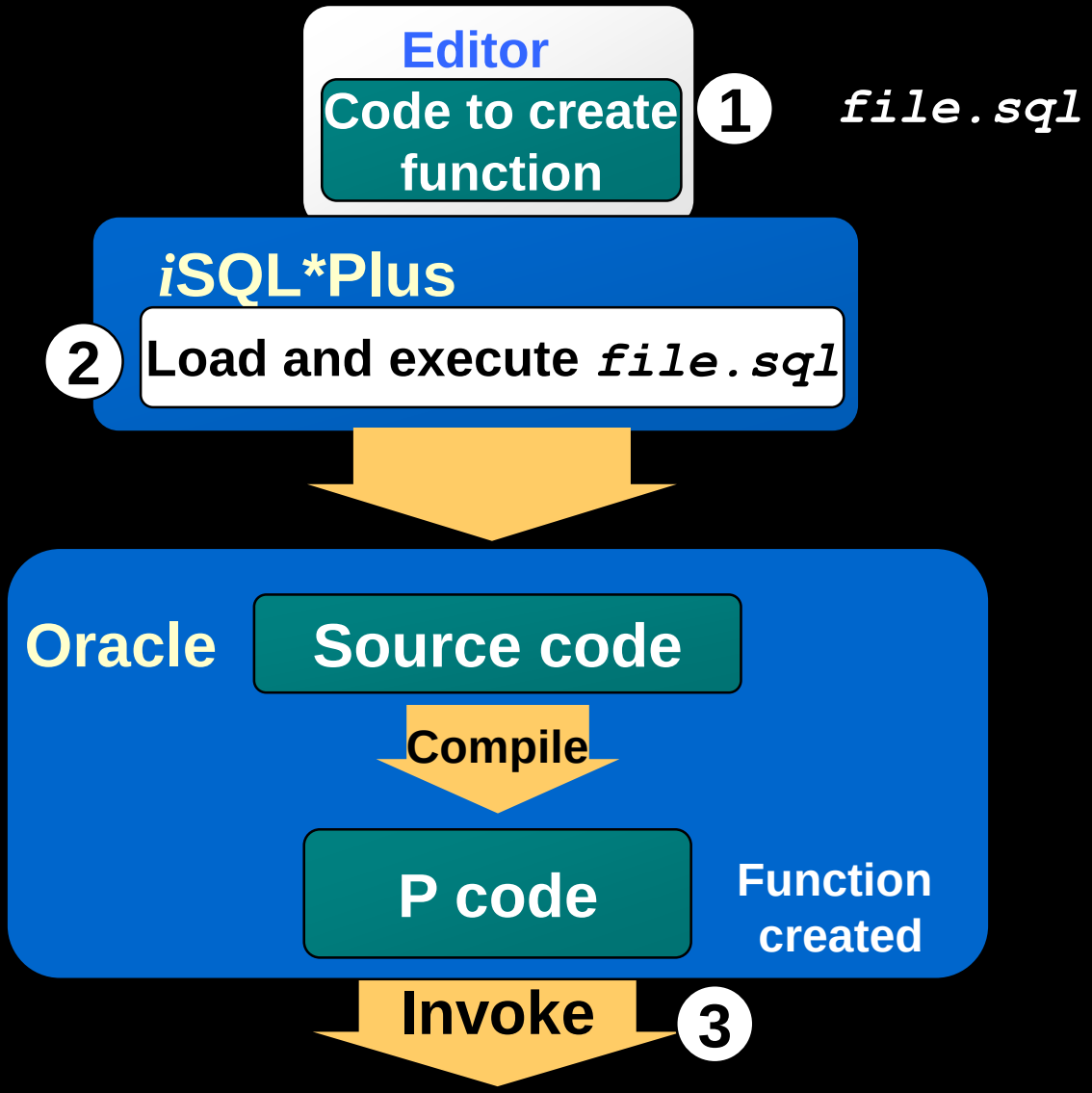
# Syntax for Creating Functions

```
CREATE [OR REPLACE] FUNCTION function_name
 [(parameter1 [mode1] datatype1,
  parameter2 [mode2] datatype2,
  . . .)]
RETURN datatype
IS|AS
PL/SQL Block;
```

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


Copyright © Oracle Corporation, 2001. All rights reserved.

# Creating a Function





Copyright © Oracle Corporation, 2001. All rights reserved.

# Creating a Stored Function by Using *i*SQL*Plus

1. Enter the text of the `CREATE FUNCTION` statement in an editor and save it as a SQL script file.
2. Run the script file to store the source code and compile the function.
3. Use `SHOW ERRORS` to see compilation errors.
4. When successfully compiled, invoke the function.


Copyright © Oracle Corporation, 2001. All rights reserved.

# Creating a Stored Function by Using *i*SQL*Plus: Example

get_salary.sql

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

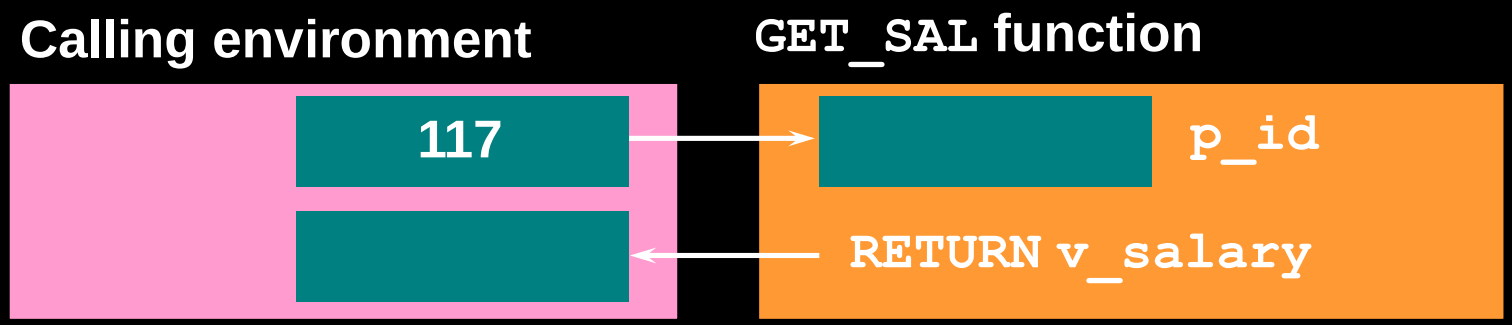

Copyright © Oracle Corporation, 2001. All rights reserved.

# Executing Functions

- Invoke a function as part of a PL/SQL expression.
- Create a variable to hold the returned value.
- Execute the function. The variable will be populated by the value returned through a `RETURN` statement.


Copyright © Oracle Corporation, 2001. All rights reserved.

# Executing Functions: Example




1. Load and run the `get_salary.sql` file to create the function

```
2  VARIABLE g_salary NUMBER

3  EXECUTE :g_salary := get_sal(117)

4  PRINT g_salary
```

```
PL/SQL procedure successfully completed.

G_SALARY
2800
```


Copyright © Oracle Corporation, 2001. All rights reserved.

# Advantages of User-Defined Functions in SQL Expressions

- Extend SQL where activities are too complex, too awkward, or unavailable with SQL
- Can increase efficiency when used in the `WHERE` clause to filter data, as opposed to filtering the data in the application
- Can manipulate character strings


Copyright © Oracle Corporation, 2001. All rights reserved.

# Invoking Functions in SQL Expressions: Example

```
CREATE OR REPLACE FUNCTION tax(p_value IN NUMBER)
 RETURN NUMBER IS
BEGIN
   RETURN (p_value * 0.08);
END tax;
/
SELECT employee_id, last_name, salary, tax(salary)
FROM   employees
WHERE  department_id = 100;
```

Function created.

| EMPLOYEE_ID | LAST_NAME | SALARY | TAX(SALARY) |
|---|---|---|---|
| 108 | Greenberg | 12000 | 960 |
| 109 | Faviet | 9000 | 720 |
| 110 | Chen | 8200 | 656 |
| 111 | Sciarra | 7700 | 616 |
| 112 | Urman | 7800 | 624 |
| 113 | Popp | 6900 | 552 |

6 rows selected.


Copyright © Oracle Corporation, 2001. All rights reserved.

# Locations to Call User-Defined Functions

- Select list of a `SELECT` command
- Condition of the `WHERE` and `HAVING` clauses
- `CONNECT BY`, `START WITH`, `ORDER BY`, and `GROUP BY` clauses
- `VALUES` clause of the `INSERT` command
- `SET` clause of the `UPDATE` command


Copyright © Oracle Corporation, 2001. All rights reserved.

# Restrictions on Calling Functions from SQL Expressions

To be callable from SQL expressions, a user-defined function must:

- Be a stored function
- Accept only `IN` parameters
- Accept only valid SQL data types, not PL/SQL specific types, as parameters
- Return data types that are valid SQL data types, not PL/SQL specific types


Copyright © Oracle Corporation, 2001. All rights reserved.

# Restrictions on Calling Functions from SQL Expressions

- Functions called from SQL expressions cannot contain DML statements.
- Functions called from `UPDATE`/`DELETE` statements on a table T cannot contain DML on the same table T.
- Functions called from an `UPDATE` or a `DELETE` statement on a table T cannot query the same table.
- Functions called from SQL statements cannot contain statements that end the transactions.
- Calls to subprograms that break the previous restriction are not allowed in the function.


Copyright © Oracle Corporation, 2001. All rights reserved.

# Restrictions on Calling from SQL

```
CREATE OR REPLACE FUNCTION dml_call_sql (p_sal NUMBER)
   RETURN NUMBER IS
BEGIN
   INSERT INTO employees(employee_id, last_name, email,
                         hire_date, job_id, salary)
        VALUES(1, 'employee 1', 'emp1@company.com',
                  SYSDATE, 'SA_MAN', 1000);
   RETURN (p_sal + 100);
END;
/
```

```
Function created.
```

```
UPDATE employees SET salary = dml_call_sql(2000)
 WHERE employee_id = 170;
```

```
UPDATE employees SET salary = dml_call_sql(2000)
                  *
ERROR at line 1:
ORA-04091: table PLSQL.EMPLOYEES is mutating, trigger/function may not see it
ORA-06512: at "PLSQL.DML_CALL_SQL", line 4
```


Copyright © Oracle Corporation, 2001. All rights reserved.

# Removing Functions

Drop a stored function.

**Syntax:**

```
DROP FUNCTION function_name
```

**Example:**

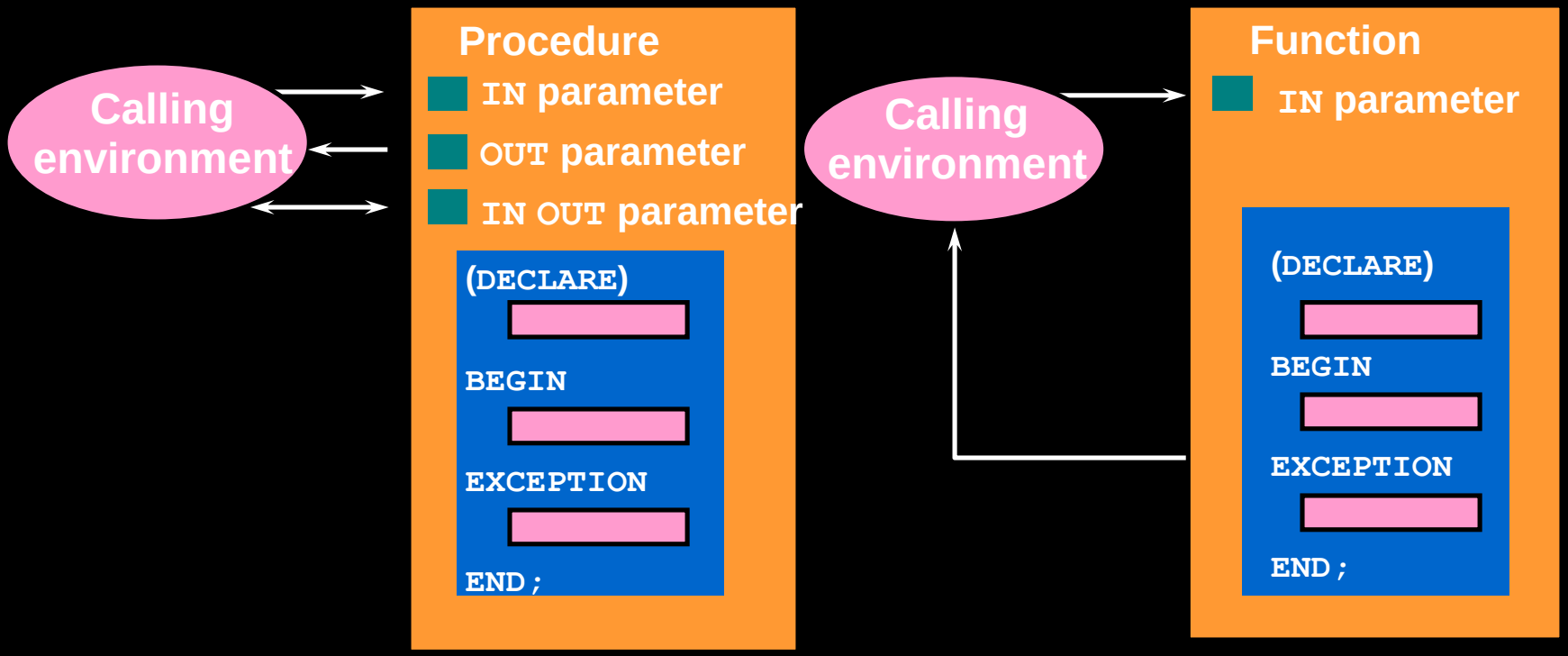
```
DROP FUNCTION get_sal;
```

```
Function dropped.
```

- All the privileges granted on a function are revoked when the function is dropped.
- The `CREATE OR REPLACE` syntax is equivalent to dropping a function and recreating it. Privileges granted on the function remain the same when this syntax is used.


Copyright © Oracle Corporation, 2001. All rights reserved.

# Procedure or Function?





Copyright © Oracle Corporation, 2001. All rights reserved.

# Comparing 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 none, one, or many values | Must return a single value |
| Can contain a `RETURN` statement | Must contain at least one `RETURN` statement |


Copyright © Oracle Corporation, 2001. All rights reserved.

# Benefits of Stored Procedures and Functions

- Improved performance
- Easy maintenance
- Improved data security and integrity
- Improved code clarity


Copyright © Oracle Corporation, 2001. All rights reserved.

# Summary

In this lesson, you should have learned that:

- A function is a named PL/SQL block that must return a value.
- A function is created by using the `CREATE FUNCTION` syntax.
- A function is invoked as part of an expression.
- A function stored in the database can be called in SQL statements.
- A function can be removed from the database by using the `DROP FUNCTION` syntax.
- Generally, you use a procedure to perform an action and a function to compute a value.


Copyright © Oracle Corporation, 2001. All rights reserved.

# Practice 10 Overview

This practice covers the following topics:

- Creating stored functions
  - To query a database table and return specific values
  - To be used in a SQL statement
  - To insert a new row, with specified parameter values, into a database table
  - Using default parameter values
- Invoking a stored function from a SQL statement
- Invoking a stored function from a stored procedure


Copyright © Oracle Corporation, 2001. All rights reserved.