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 database object, for repeated execution.
- A function can be called as part of an expression.

Syntax for Creating Functions

```
CREATE [OR REPLACE] FUNCTION function_name
  (argument1 [mode1] datatype1,
   argument2 [mode2] datatype2,
   . . .
RETURN datatype
IS|AS
PL/SQL Block;
```

Creating a Stored Function Using SQL*Plus

1. Enter the text of the CREATE FUNCTION statement into the Sql Worksheet of SqlDeveloper.
2. From SqlDeveloper, run the script to compile the source code into p-code and store both in the database.
3. Invoke the function from an Oracle Server environment to determine whether it executes without error.

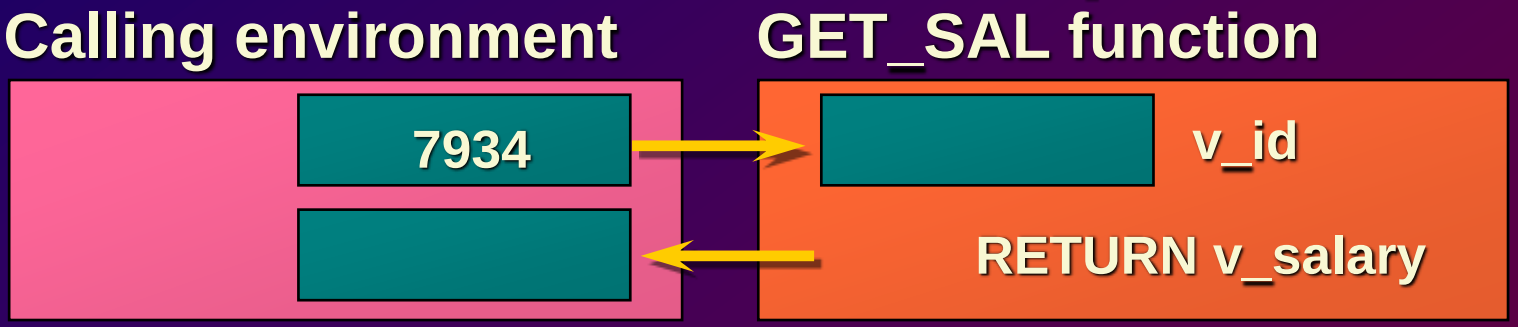
Creating a Stored Function Using SQL*Plus: Example

```
SQL> CREATE OR REPLACE FUNCTION get_sal
  2    (v_id  IN   emp.empno%TYPE)
  3  RETURN NUMBER
  4  IS
  5    v_salary      emp.sal%TYPE :=0;
  6  BEGIN
  7    SELECT sal
  8    INTO   v_salary
  9    FROM   emp
 10    WHERE  empno = v_id;
 11    RETURN (v_salary);
 12 END get_sal;
 13 /
```

Executing Functions

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

Executing Functions in SQL*Plus: Example




```
SQL> START get_salary.sql
Procedure created.
```

```
SQL> VARIABLE g_salary number
```

```
SQL> EXECUTE :g_salary := get_sal(7934)
PL/SQL procedure successfully completed.
```

```
SQL> PRINT g_salary
          G_SALARY
-----------------
              1300
```

Advantages of User-Defined Functions in SQL Expressions

- **Extend SQL** where activities are too complex, too awkward, or unavailable with SQL
- Query efficiency: functions used in the WHERE clause can filter data
- Manipulate character strings
- Provide parallel query execution

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 clauses of the INSERT command
- SET clause of the UPDATE command

Calling Functions from SQL Expressions: Restrictions

- A user-defined function must be a stored function.
- A user-defined function must be a ROW function, not a GROUP function.
- A user-defined function only takes IN parameters, not OUT, or IN OUT.
- Datatypes must be CHAR, DATE, or NUMBER, not PL/SQL types such as BOOLEAN, RECORD, or TABLE.
- Return type must be an Oracle Server internal type.

Calling Functions from SQL Expressions: Restrictions

- INSERT, UPDATE, or DELETE commands are not allowed.
- Calls to subprograms that break the above restriction are not allowed.

Removing a Server-Side Function

Using SQL*Plus

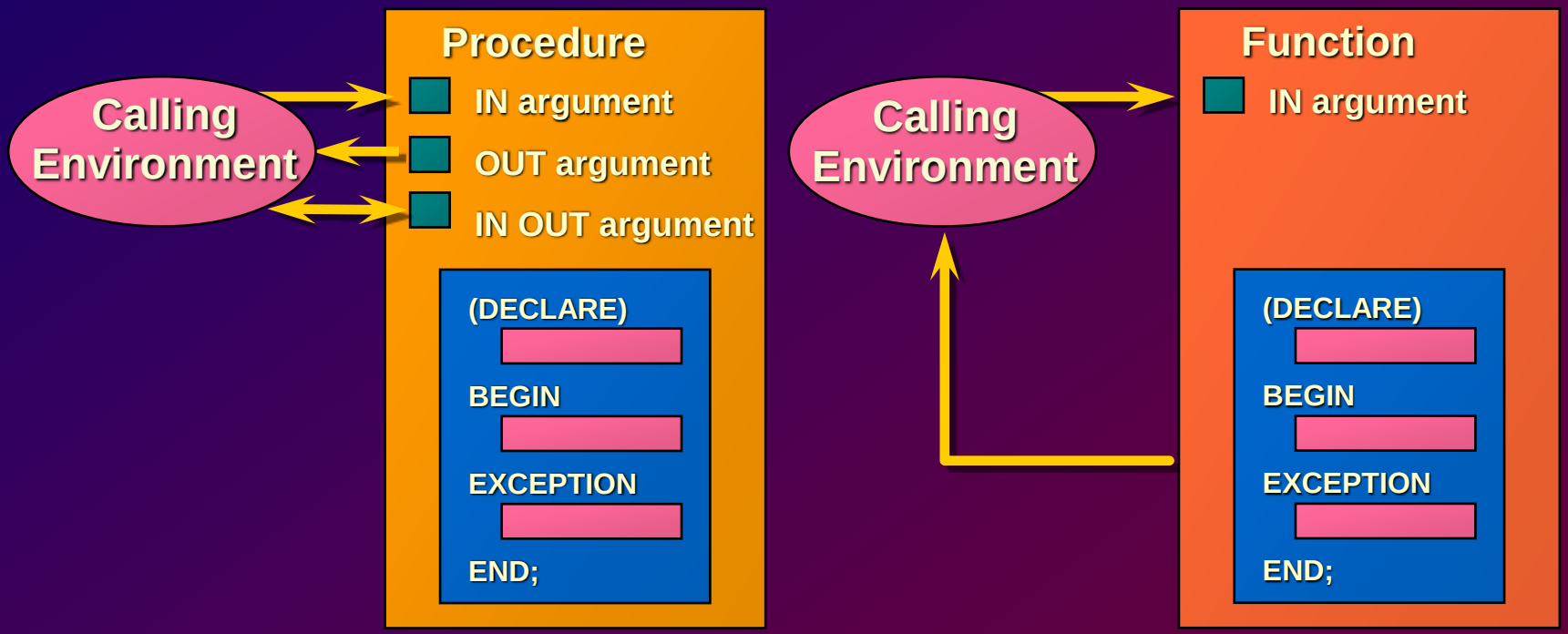
- Syntax

```
DROP FUNCTION function_name
```

- Example

```
SQL> DROP FUNCTION get_salary;
Function dropped.
```

Procedure or Function?



Comparing Procedures and Functions

Procedure	Function
Execute as a PL/SQL statement	Invoke as part of an expression
No RETURN datatype	Must contain a RETURN datatype
Can return one or more values	Must return a value

Benefits of Stored Procedures and Functions

- Improved performance
- Improved maintenance
- Improved data security and integrity