Overview of Procedures

- A procedure is a named PL/SQL block that performs an action.
- A procedure can be stored in the database, as a database object, for repeated execution.

Syntax for Creating Procedures

```
CREATE [OR REPLACE] PROCEDURE procedure_name
  (argument1 [mode1] datatype1,
   argument2 [mode2] datatype2,
   . . .
IS [AS]
PL/SQL Block;
```

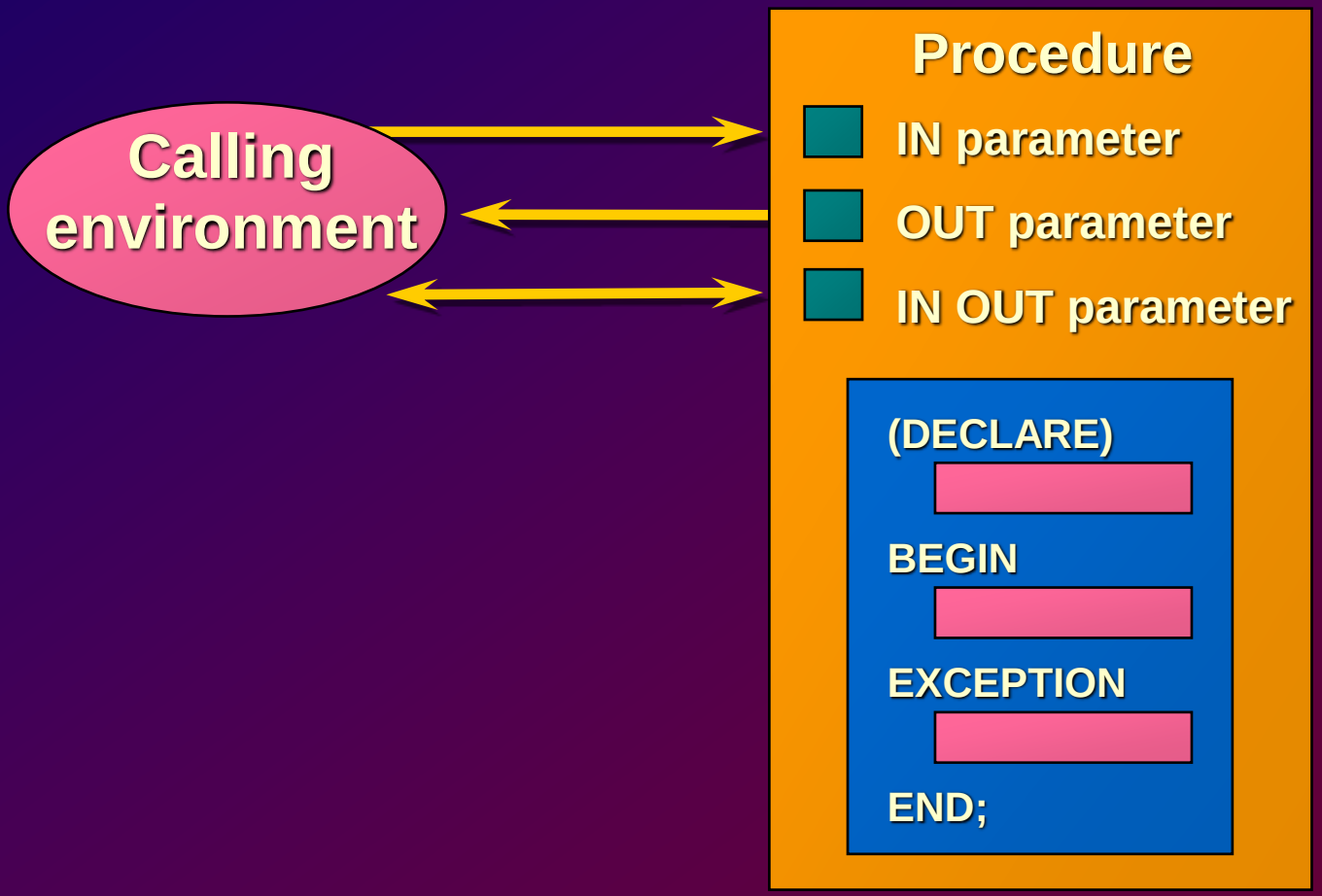
Creating a Stored Procedure Using SQL*Plus

1. Enter the text of the CREATE PROCEDURE 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 procedure from an Oracle Server environment to determine whether it executes without error.

Procedural Parameter Modes



Parameter Modes for Formal Parameters

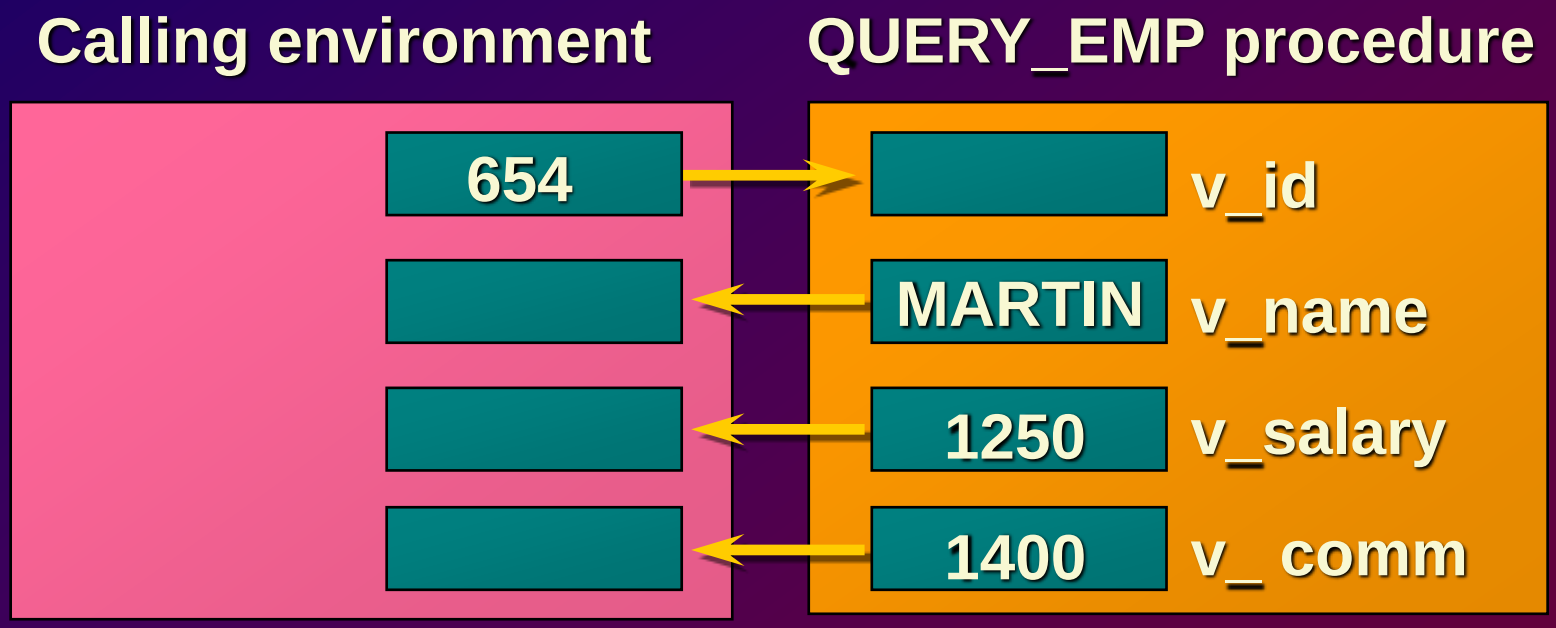
IN	OUT	IN OUT
Default	Must be specified	Must be specified
Value is passed into subprogram	Returned to calling environment	Passed into subprogram; returned to calling environment
Formal parameter acts as a constant	Uninitialized variable	Initialized variable
Actual parameter can be a literal, expression, constant, or initialized variable	Must be a variable	Must be a variable

IN Parameters: Example


```
SQL> CREATE OR REPLACE PROCEDURE raise_salary
  2   (v_id in emp.empno%TYPE)
  3   IS
  4   BEGIN
  5       UPDATE emp
  6       SET    sal = sal * 1.10
  7       WHERE  empno = v_id;
  8   END raise_salary;
  9   /
Procedure created.

SQL> EXECUTE raise_salary (7369)
PL/SQL procedure successfully completed.
```

OUT Parameters: Example



OUT Parameters: Example

```
SQL> CREATE OR REPLACE PROCEDURE query_emp
  1  (v_id      IN   emp.empno%TYPE,
  2   v_name    OUT  emp.ename%TYPE,
  3   v_salary OUT   emp.sal%TYPE,
  4   v_comm    OUT  emp.comm%TYPE)
  5  IS
  6  BEGIN
  7    SELECT   ename, sal, comm
  8    INTO     v_name, v_salary, v_comm
  9    FROM     emp
 10    WHERE    empno = v_id;
 11  END query_emp;
 12  /
```

OUT Parameters and SQL*Plus

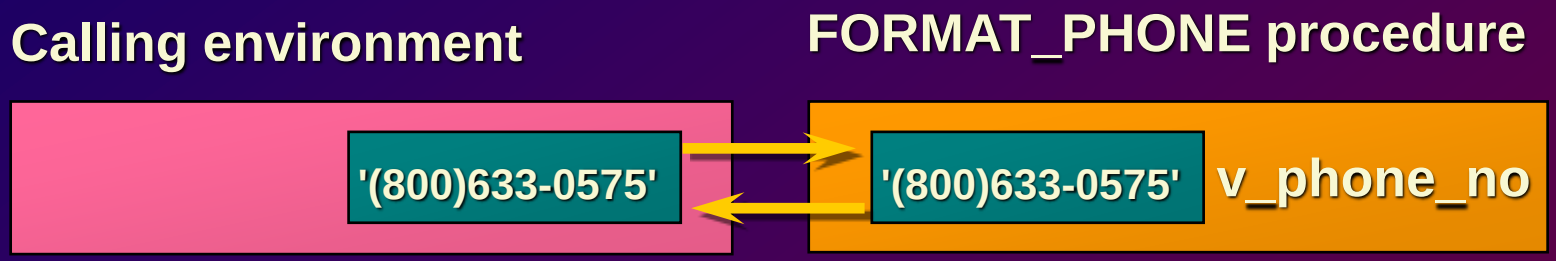
```
SQL> START emp_query.sql
Procedure created.
```

```
SQL> VARIABLE g_namevarchar2(15)
SQL> VARIABLE g_salary     number
SQL> VARIABLE g_commnumber
```

```
SQL> EXECUTE query_emp (7654, :g_name, :g_salary,
  2   :g_comm)
PL/SQL procedure successfully completed.
```

```
SQL> PRINT g_name
G_NAME
---------------
MARTIN
```

IN OUT Parameters




```
SQL> CREATE OR REPLACE PROCEDURE format_phone
  2   (v_phone_no IN OUT VARCHAR2)
  3   IS
  4   BEGIN
  5    v_phone_no := '(' || SUBSTR(v_phone_no,1,3) ||
  6                  ')' || SUBSTR(v_phone_no,4,3) ||
  7                  '-' || SUBSTR(v_phone_no,7);
  8   END format_phone;
  9 /
```

Invoking FORMAT_PHONE from SQL*Plus

```
SQL>VARIABLE g_phone_no varchar2(15)

SQL> BEGIN :g_phone_no := '8006330575'; END;
  2  /
PL/SQL procedure successfully completed.

SQL> EXECUTE format_phone (:g_phone_no)
PL/SQL procedure successfully completed.

SQL> PRINT g_phone_no
```

```
G_PHONE_NO
---------------
(800)633-0575
```

Methods for Passing Parameters

- Positional
- Named
- Combination

Passing Parameters: Example Procedure

```
SQL> CREATE OR REPLACE PROCEDURE add_dept
  1   (v_name   IN dept.dname%TYPE   DEFAULT 'unknown',
  2    v_loc    IN dept.loc%TYPE     DEFAULT 'unknown')
  3  IS
  4  BEGIN
  5     INSERT INTO dept
  6     VALUES (dept_deptno.NEXTVAL, v_name, v_loc);
  7  END add_dept;
  8  /
```

Examples of Passing Parameters

```
SQL> begin
  2  add_dept;
  3  add_dept ( 'TRAINING', 'NEW YORK');
  4  add_dept ( v_loc => 'DALLAS', v_name =>
                'EDUCATION') ;
  5  add_dept ( v_loc => 'BOSTON') ;
  6  end;
  7  /
PL/SQL procedure successfully completed.
```

```
SQL>      SELECT * FROM dept;

DEPTNO    DNAME              LOC
------    --------------     -------------
   ...    ...                ...
    41    unknown            unknown
    42    TRAINING           NEW YORK
    43    EDUCATION          DALLAS
    44    unknown            BOSTON
```

Invoking a Procedure from an Anonymous PL/SQL Block

```
DECLARE
  v_id NUMBER := 7900;
BEGIN
  raise_salary(v_id);     --invoke procedure
COMMIT;
...

END;
```

Invoking a Procedure from a Stored Procedure

```
SQL> CREATE OR REPLACE PROCEDURE process_emps
  2  IS
  3    CURSOR emp_cursor IS
  4    SELECT empno
  5    FROM   emp;
  6  BEGIN
  7    FOR emp_rec IN emp_cursor LOOP
  8       raise_salary(emp_rec.empno);  --invoke procedure
  9    END LOOP;
 10  COMMIT;
 11  END process_emps;
 12 /
```