Creating Procedures

Copyright © Oracle Corporation, 2001. All rights reserved.

Objectives

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

- Distinguish anonymous PL/SQL blocks from named PL/SQL blocks (subprograms)
- Describe subprograms
- List the benefits of using subprograms
- List the different environments from which subprograms can be invoked

Copyright © Oracle Corporation, 2001. All rights reserved.

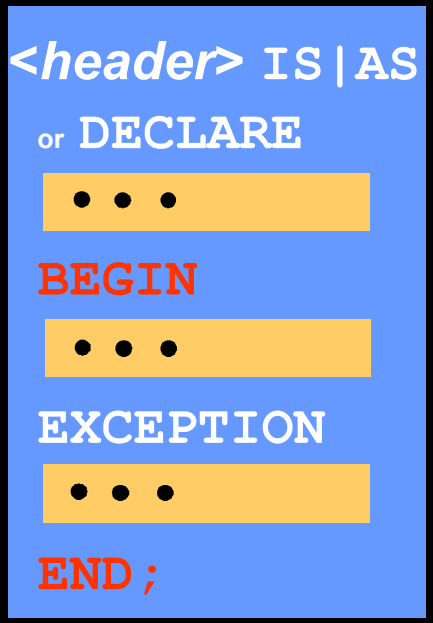
Objectives

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

- Describe PL/SQL blocks and subprograms
- Describe the uses of procedures
- Create procedures
- Differentiate between formal and actual parameters
- List the features of different parameter modes
- Create procedures with parameters
- Invoke a procedure
- Handle exceptions in procedures
- Remove a procedure

Copyright © Oracle Corporation, 2001. All rights reserved.

PL/SQL Program Constructs



Tools Constructs
Anonymous blocks
Application procedures or functions
Application packages
Application triggers
Object types

Database Server Constructs
Anonymous blocks
Stored procedures or functions
Stored packages
Database triggers
Object types

Copyright © Oracle Corporation, 2001. All rights reserved.

Overview of Subprograms

A subprogram:

- Is a named PL/SQL block that can accept parameters and be invoked from a calling environment
- Is of two types:
  - A procedure that performs an action
  - A function that computes a value
- Is based on standard PL/SQL block structure
- Provides modularity, reusability, extensibility, and maintainability
- Provides easy maintenance, improved data security and integrity, improved performance, and improved code clarity

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

Block Structure for Anonymous PL/SQL Blocks

DECLARE (optional)
Declare PL/SQL objects to be used within this block

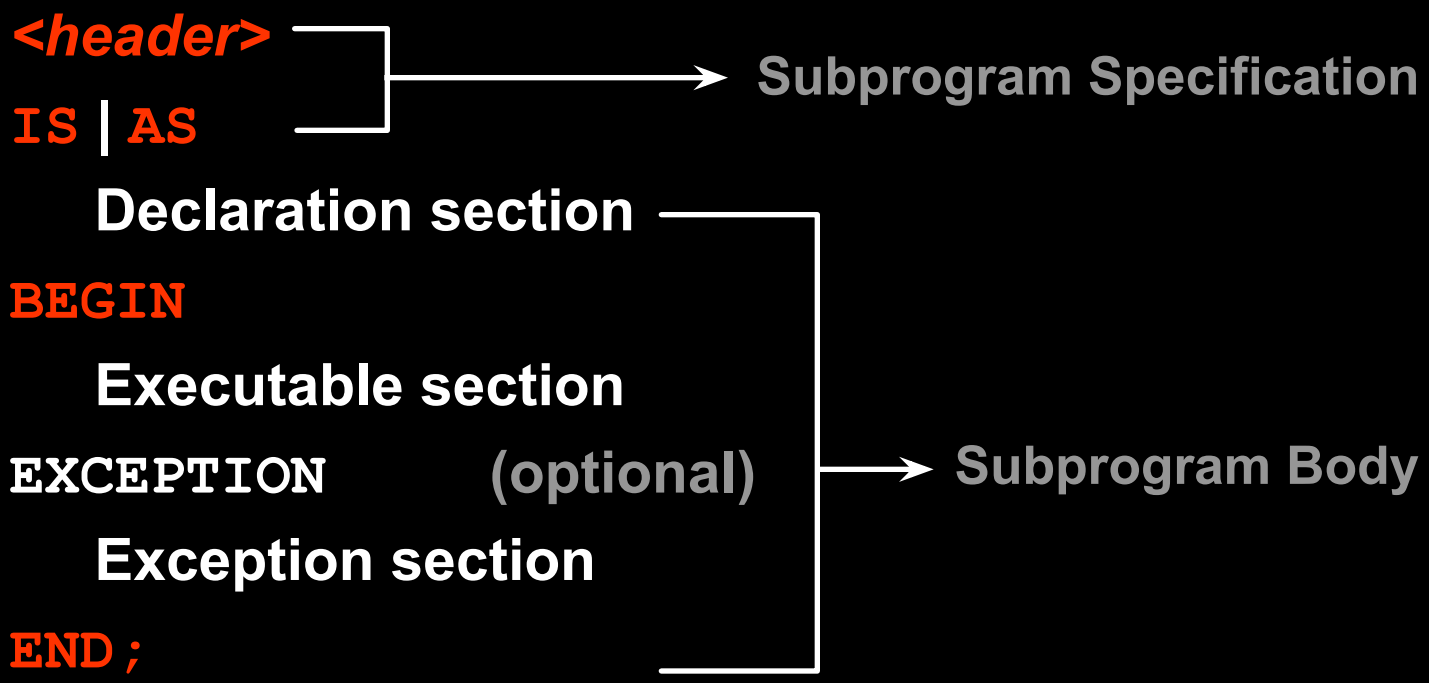
BEGIN (mandatory)
Define the executable statements

EXCEPTION (optional)
Define the actions that take place if an error or exception arises

END; (mandatory)

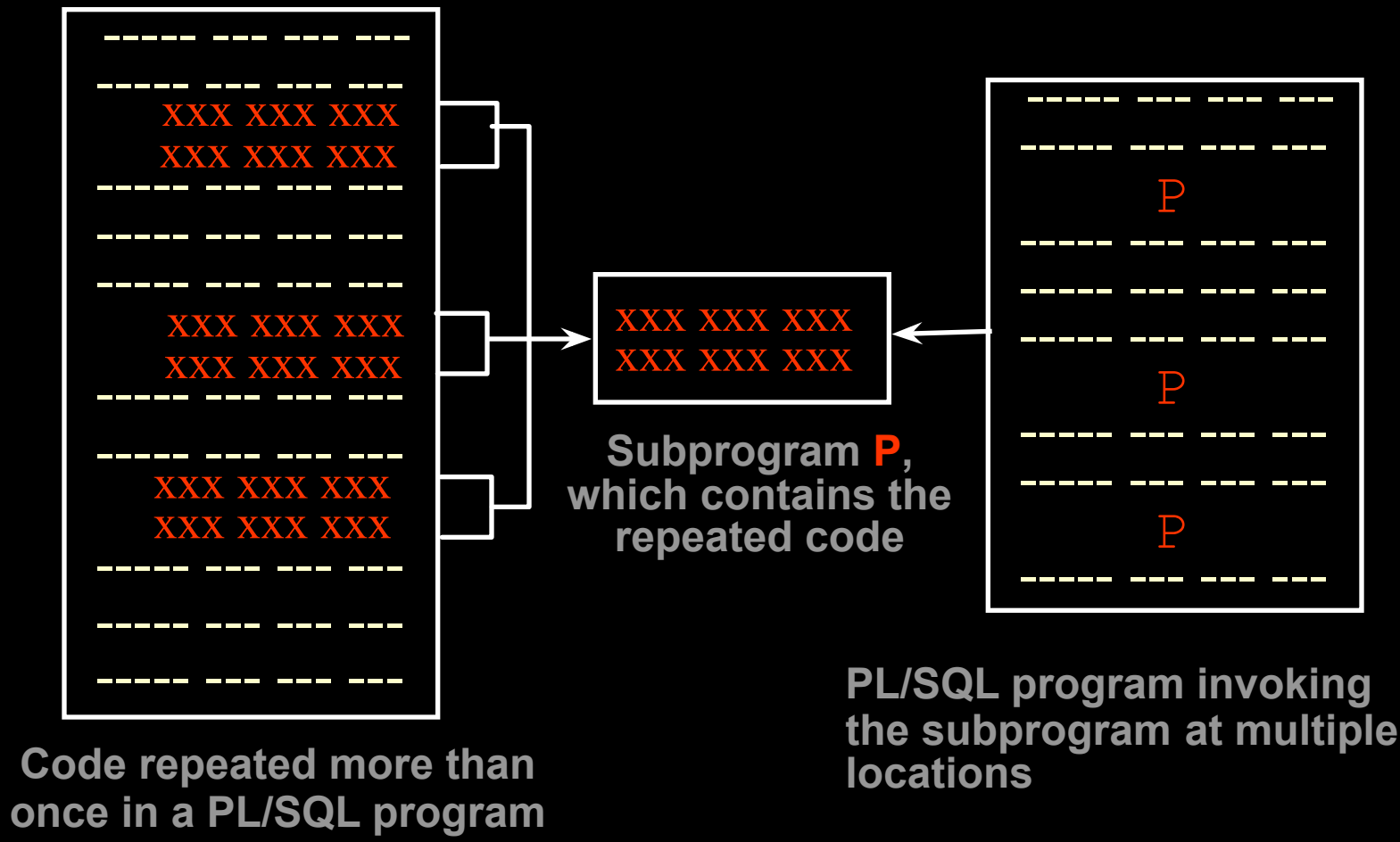
Copyright © Oracle Corporation, 2001. All rights reserved.

Block Structure for PL/SQL Subprograms



Copyright © Oracle Corporation, 2001. All rights reserved.

PL/SQL Subprograms



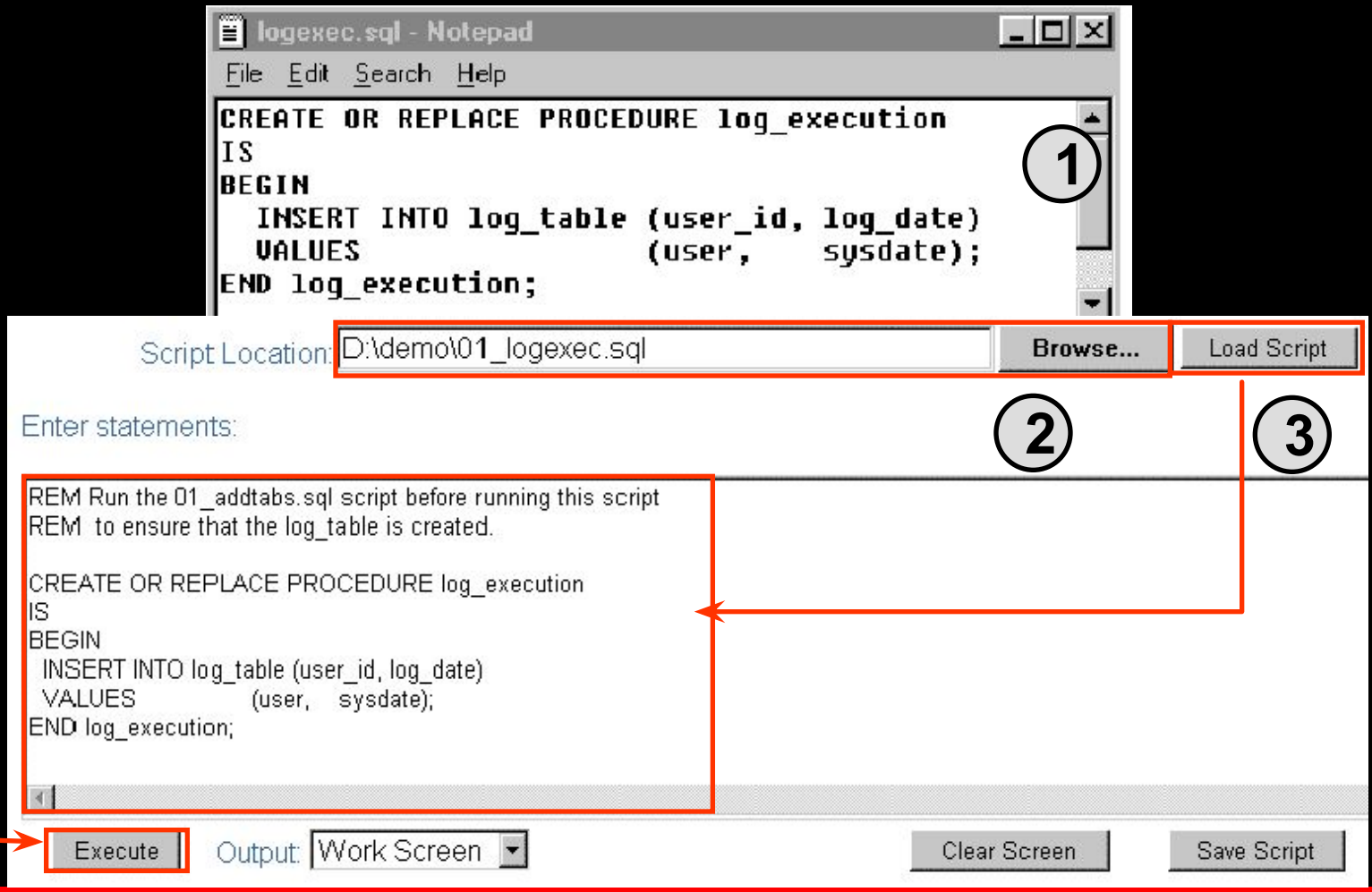
Copyright © Oracle Corporation, 2001. All rights reserved.

Benefits of Subprograms

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

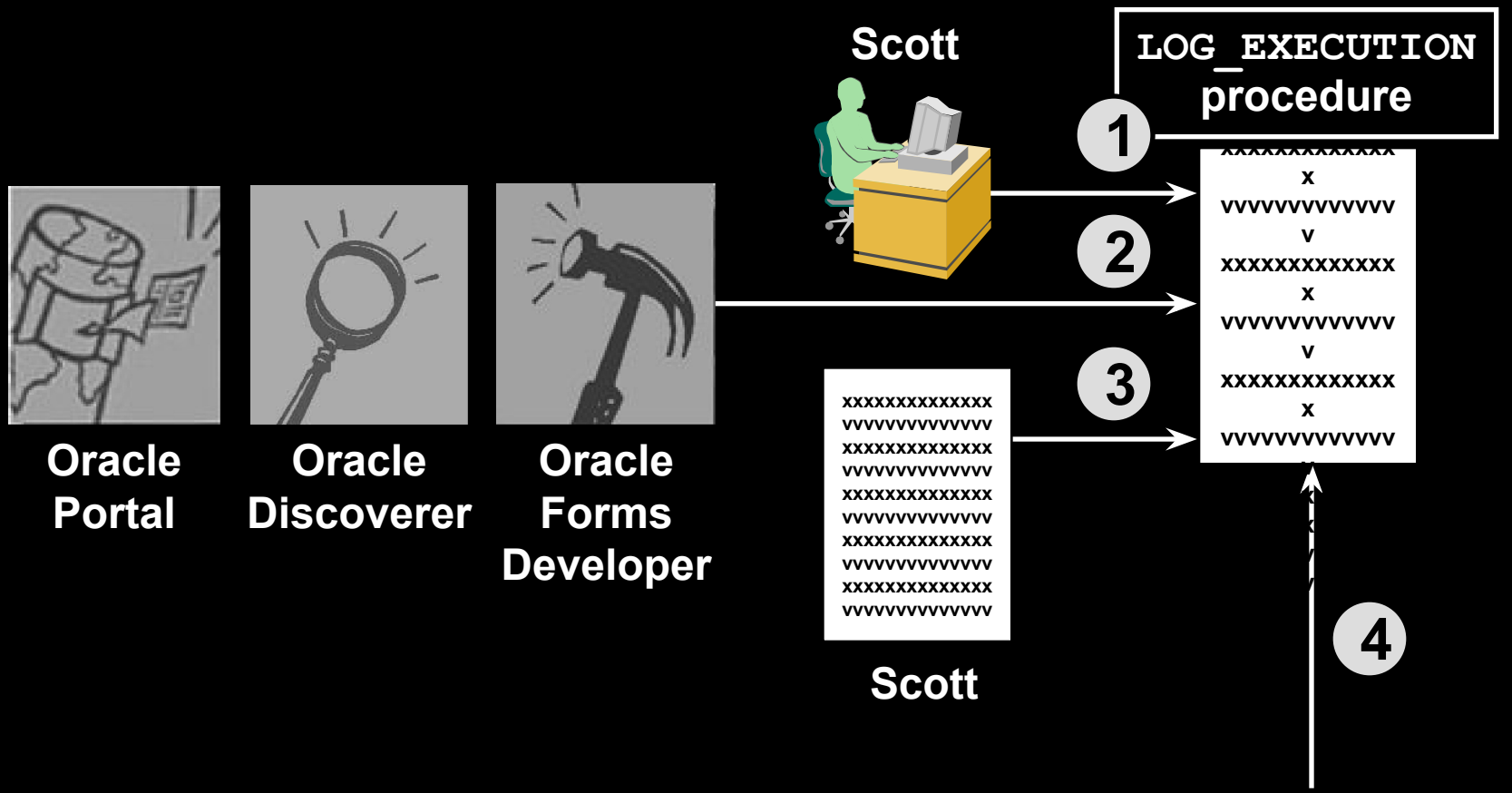
Copyright © Oracle Corporation, 2001. All rights reserved.

Developing Subprograms by Using *i*SQL*Plus



Copyright © Oracle Corporation, 2001. All rights reserved.

Invoking Stored Procedures and Functions



Copyright © Oracle Corporation, 2001. All rights reserved.

What Is a Procedure?

- A procedure is a type of subprogram that performs an action.
- A procedure can be stored in the database, as a schema object, for repeated execution.

Copyright © Oracle Corporation, 2001. All rights reserved.

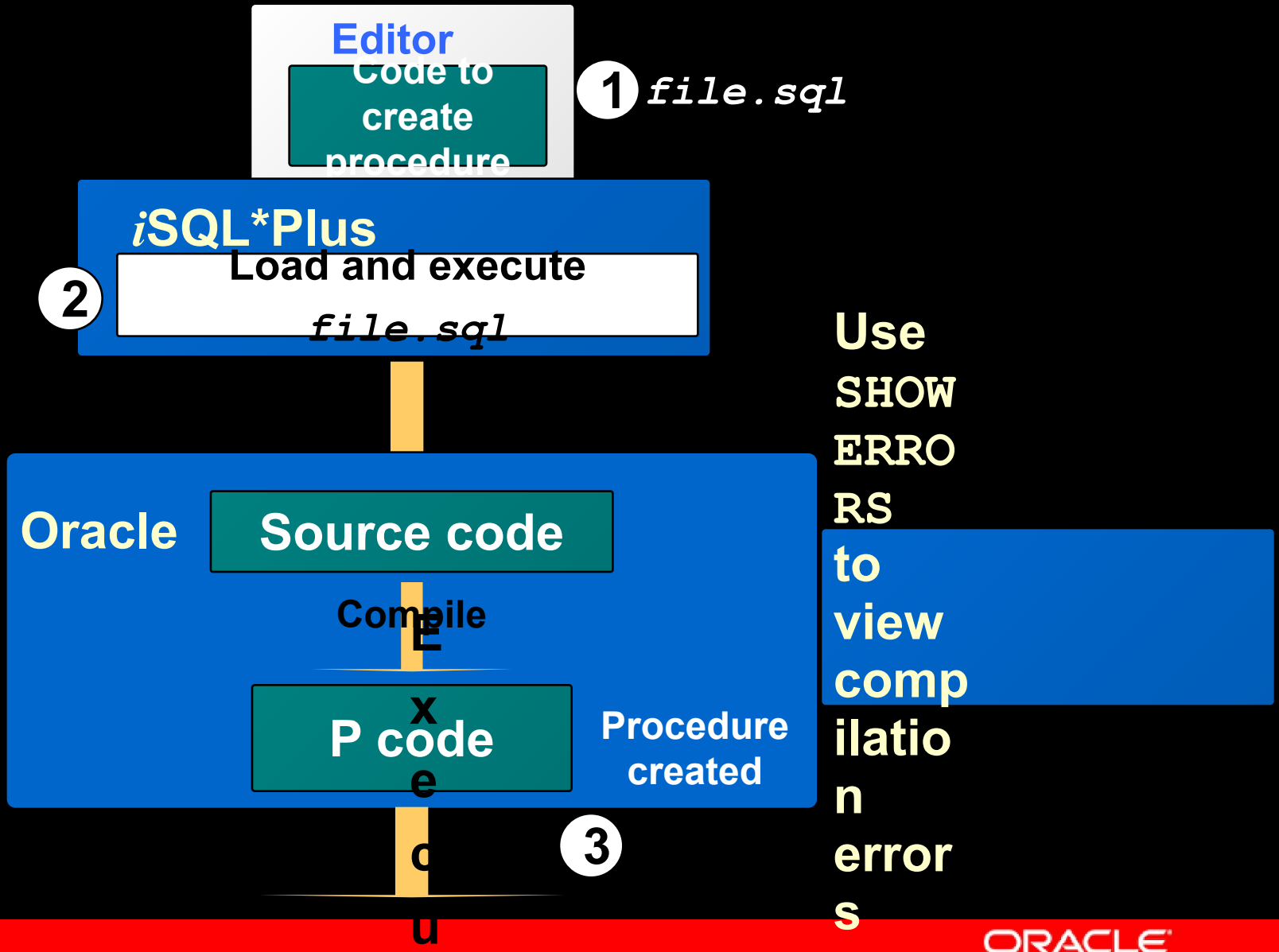
Syntax for Creating Procedures

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

- The `REPLACE` option indicates that if the procedure exists, it will be dropped and replaced with the new version created by the statement.
- PL/SQL block starts with either `BEGIN` or the declaration of local variables and ends with either `END` or `END` *procedure_name.*

Copyright © Oracle Corporation, 2001. All rights reserved.

Developing Procedures



Copyright © Oracle Corporation, 2001. All rights reserved.

Formal Versus Actual Parameters

- Formal parameters: variables declared in the parameter list of a subprogram specification

  Example:

```
CREATE PROCEDURE raise_sal(p_id NUMBER, p_amount NUMBER)
...
END raise_sal;
```

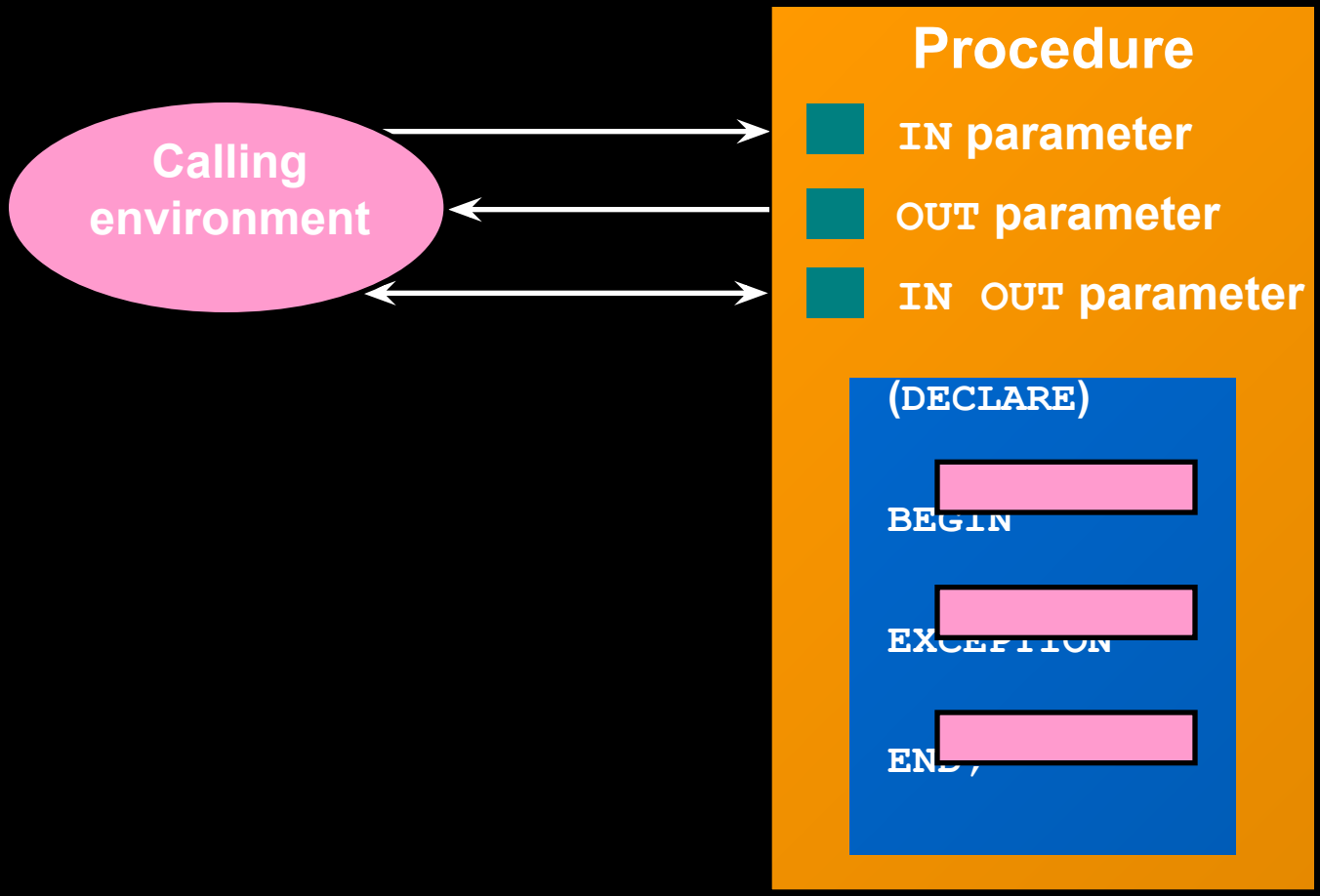
- Actual parameters: variables or expressions referenced in the parameter list of a subprogram call

  Example:

```
raise_sal(v_id, 2000)
```


Copyright © Oracle Corporation, 2001. All rights reserved.

Procedural Parameter Modes



Copyright © Oracle Corporation, 2001. All rights reserved.

Creating Procedures with Parameters

IN	OUT	IN OUT
Default mode	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
Can be assigned a default value	Cannot be assigned a default value	Cannot be assigned a default value

Copyright © Oracle Corporation, 2001. All rights reserved.

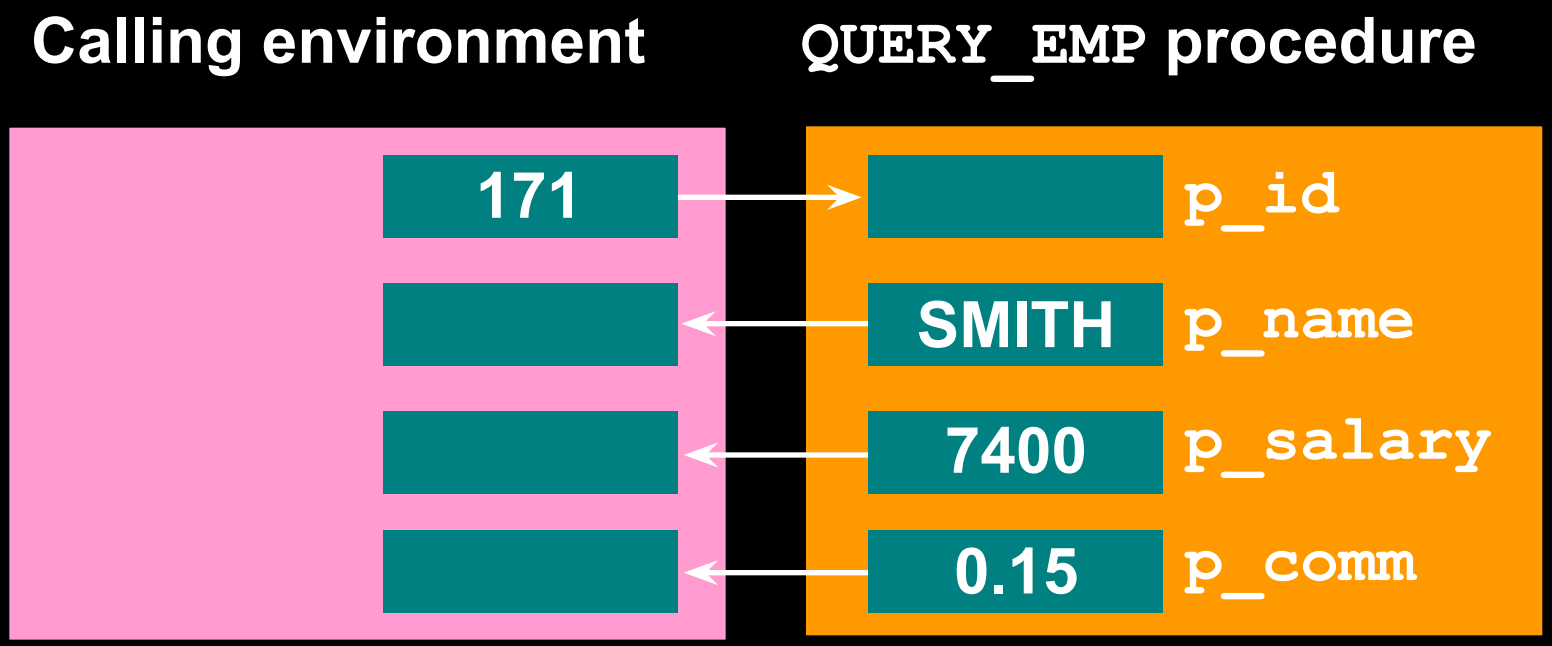
IN Parameters: Example


```
CREATE OR REPLACE PROCEDURE raise_salary
   (p_id IN employees.employee_id%TYPE)
IS
BEGIN
   UPDATE employees
   SET    salary = salary * 1.10
   WHERE  employee_id = p_id;
END raise_salary;
/
```

```
Procedure created.
```


Copyright © Oracle Corporation, 2001. All rights reserved.

OUT Parameters: Example



Copyright © Oracle Corporation, 2001. All rights reserved.

OUT Parameters: Example

emp_query.sql

```
CREATE OR REPLACE PROCEDURE query_emp
   (p_id       IN   employees.employee_id%TYPE,
    p_name     OUT  employees.last_name%TYPE,
    p_salary   OUT  employees.salary%TYPE,
    p_comm     OUT  employees.commission_pct%TYPE)
IS
BEGIN
  SELECT   last_name, salary, commission_pct
    INTO    p_name, p_salary, p_comm
    FROM    employees
    WHERE   employee_id = p_id;
END query_emp;
/
```

```
Procedure created.
```


Copyright © Oracle Corporation, 2001. All rights reserved.

Viewing OUT Parameters

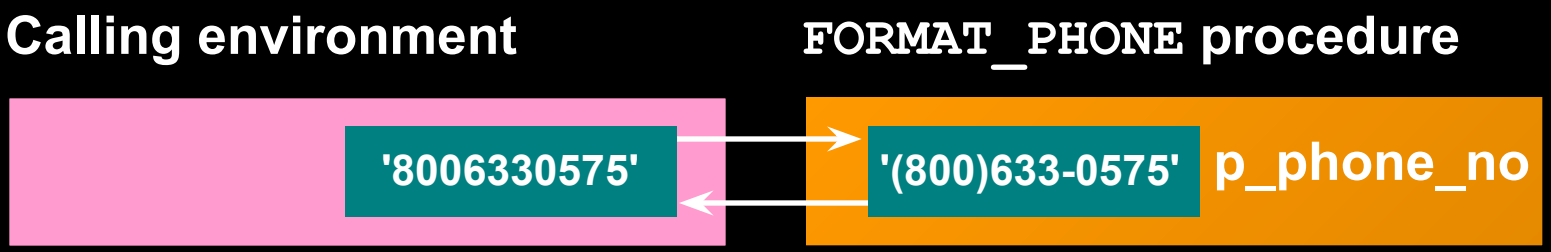
- Load and run the `emp_query.sql` script file to create the `QUERY_EMP` procedure.
- Declare host variables, execute the `QUERY_EMP` procedure, and print the value of the global `G_NAME` variable

```
VARIABLE g_name  VARCHAR2(25)
VARIABLE g_sal   NUMBER
VARIABLE g_comm  NUMBER

EXECUTE query_emp(171, :g_name, :g_sal, :g_comm)

PRINT g_name
```

```
PL/SQL procedure successfully completed.

G_NAME
Smith
```


Copyright © Oracle Corporation, 2001. All rights reserved.

IN OUT Parameters




```
CREATE OR REPLACE PROCEDURE format_phone
   (p_phone_no IN OUT VARCHAR2)
IS
BEGIN
  p_phone_no := '(' || SUBSTR(p_phone_no,1,3) ||
                ')' || SUBSTR(p_phone_no,4,3) ||
                '-' || SUBSTR(p_phone_no,7);
END format_phone;
/
```

```
Procedure created.
```


Copyright © Oracle Corporation, 2001. All rights reserved.

Viewing IN OUT Parameters

```
VARIABLE g_phone_no VARCHAR2(15)
BEGIN
  :g_phone_no := '8006330575';
END;
/
PRINT g_phone_no
EXECUTE format_phone (:g_phone_no)
PRINT g_phone_no
```

PL/SQL procedure successfully completed.

G_PHONE_NO
8006330575

PL/SQL procedure successfully completed.

G_PHONE_NO
(800)633-0575

Copyright © Oracle Corporation, 2001. All rights reserved.

Methods for Passing Parameters

- Positional: List actual parameters in the same order as formal parameters.
- Named: List actual parameters in arbitrary order by associating each with its corresponding formal parameter.
- Combination: List some of the actual parameters as positional and some as named.

Copyright © Oracle Corporation, 2001. All rights reserved.

DEFAULT Option for Parameters

```
CREATE OR REPLACE PROCEDURE add_dept
   (p_name  IN departments.department_name%TYPE
               DEFAULT 'unknown',
    p_loc   IN departments.location_id%TYPE
               DEFAULT 1700)
IS
BEGIN
   INSERT INTO departments(department_id,
               department_name, location_id)
   VALUES (departments_seq.NEXTVAL, p_name, p_loc);
END add_dept;
/
```

```
Procedure created.
```


Copyright © Oracle Corporation, 2001. All rights reserved.

Examples of Passing Parameters

```
BEGIN
  add_dept;
  add_dept ('TRAINING', 2500);
  add_dept ( p_loc => 2400, p_name
=>'EDUCATION');
  add_dept ( p_loc => 1200) ;
END;
/
SELECT department_id, department_name,
```

PL/SQL procedure successfully completed.

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
30	Purchasing	1700
40	Human Resources	2400
...		
290	TRAINING	2500
300	EDUCATION	2400
310	unknown	1200

31 rows selected.

Copyright © Oracle Corporation, 2001. All rights reserved.

Declaring Subprograms

leave_emp2.sql

```
CREATE OR REPLACE PROCEDURE leave_emp2
   (p_id  IN  employees.employee_id%TYPE)
IS
  PROCEDURE log_exec
  IS
  BEGIN
    INSERT INTO log_table (user_id, log_date)
    VALUES (USER, SYSDATE);
  END log_exec;
BEGIN
  DELETE FROM employees
  WHERE employee_id = p_id;
  log_exec;
END leave_emp2;
/
```


Copyright © Oracle Corporation, 2001. All rights reserved.

Invoking a Procedure from an Anonymous PL/SQL Block

```
DECLARE
  v_id NUMBER := 163;
BEGIN
  raise_salary(v_id);    --invoke
procedure
  COMMIT;
...
```


Copyright © Oracle Corporation, 2001. All rights reserved.

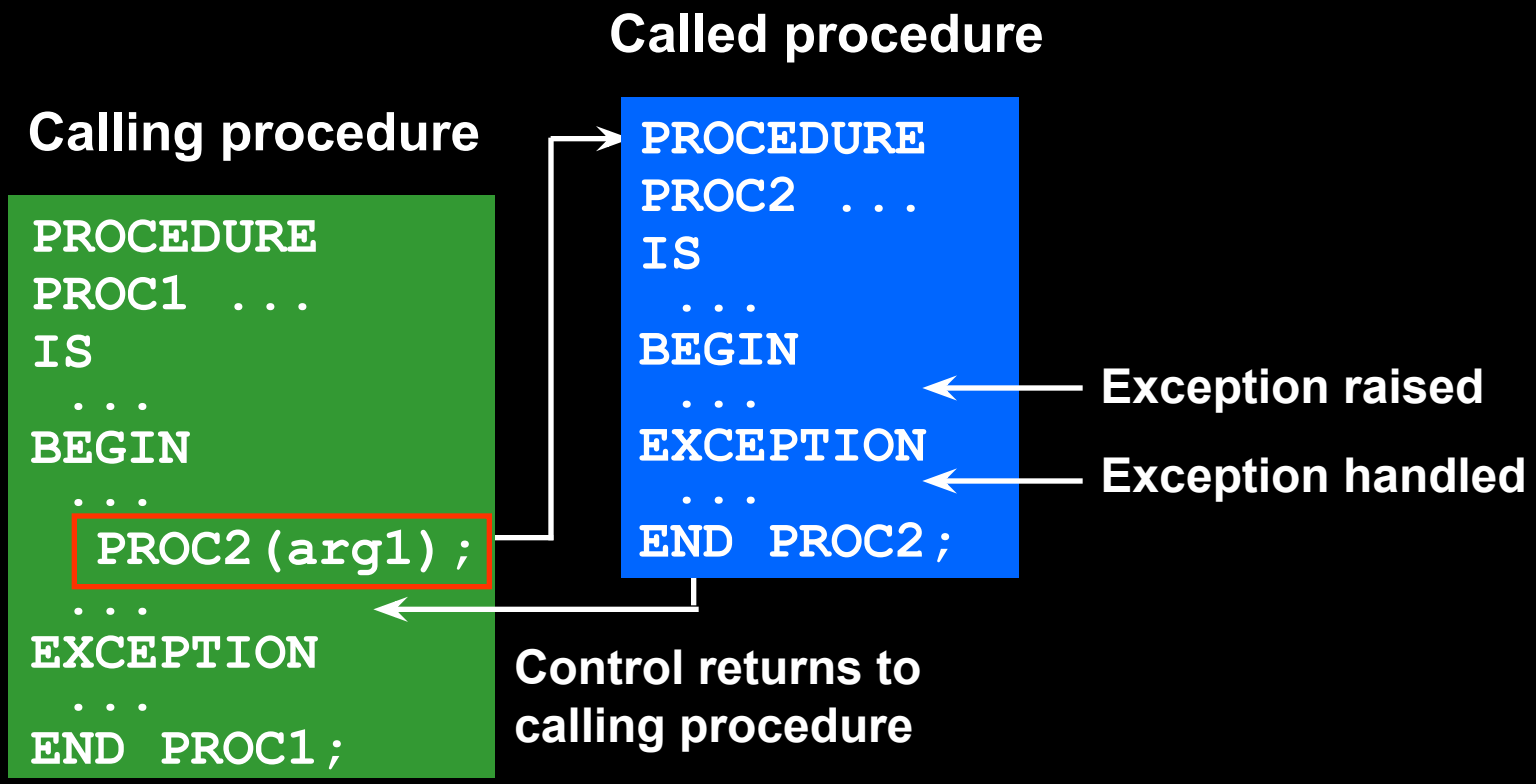
Invoking a Procedure from Another Procedure

process_emps.sql

```
CREATE OR REPLACE PROCEDURE
process_emps
IS
    CURSOR emp_cursor IS
     SELECT employee_id
     FROM   employees;
BEGIN
    FOR emp_rec IN emp_cursor
    LOOP
     raise_salary(emp_rec.employee_id);
    END LOOP;
    COMMIT;
END process_emps;
```


Copyright © Oracle Corporation, 2001. All rights reserved.

Handled Exceptions



Copyright © Oracle Corporation, 2001. All rights reserved.

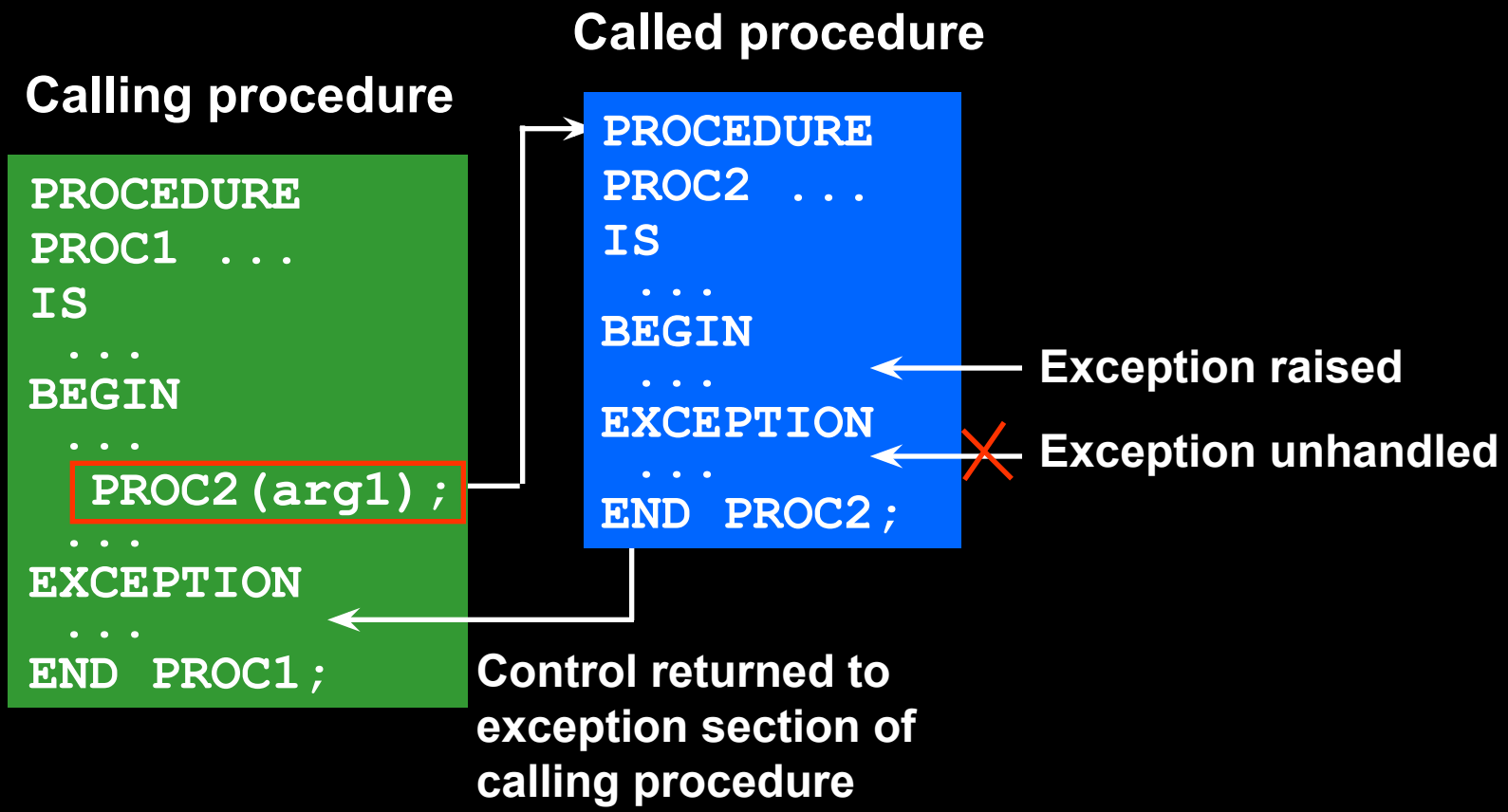
Handled Exceptions

```
CREATE PROCEDURE p2_ins_dept(p_locid NUMBER) IS
 v_did NUMBER(4);
BEGIN
 DBMS_OUTPUT.PUT_LINE('Procedure p2_ins_dept started');
 INSERT INTO departments VALUES (5, 'Dept 5', 145, p_locid);
 SELECT department_id INTO v_did FROM employees
  WHERE employee_id = 999;
END;
```

```
CREATE PROCEDURE p1_ins_loc(p_lid NUMBER, p_city VARCHAR2)
IS
 v_city VARCHAR2(30); v_dname VARCHAR2(30);
BEGIN
 DBMS_OUTPUT.PUT_LINE('Main Procedure p1_ins_loc');
 INSERT INTO locations (location_id, city) VALUES (p_lid, p_city);
 SELECT city INTO v_city FROM locations WHERE location_id = p_lid;
 DBMS_OUTPUT.PUT_LINE('Inserted city '||v_city);
 DBMS_OUTPUT.PUT_LINE('Invoking the procedure p2_ins_dept ...');
 p2_ins_dept(p_lid);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
   DBMS_OUTPUT.PUT_LINE('No such dept/loc for any employee');
END;
```


Copyright © Oracle Corporation, 2001. All rights reserved.

Unhandled Exceptions



Copyright © Oracle Corporation, 2001. All rights reserved.

Unhandled Exceptions

```
CREATE PROCEDURE p2_noexcep(p_locid NUMBER) IS
 v_did NUMBER(4);
BEGIN
 DBMS_OUTPUT.PUT_LINE('Procedure p2_noexcep started');
 INSERT INTO departments VALUES (6, 'Dept 6', 145, p_locid);
 SELECT department_id INTO v_did FROM employees
  WHERE employee_id = 999;
END;
```

```
IS
 v_city VARCHAR2(30); v_dname VARCHAR2(30);
BEGIN
 DBMS_OUTPUT.PUT_LINE(' Main Procedure p1_noexcep');
 INSERT INTO locations (location_id, city) VALUES (p_lid,
p_city);
 SELECT city INTO v_city FROM locations WHERE location_id =
p_lid;
 DBMS_OUTPUT.PUT_LINE('Inserted new city '||v_city);
 DBMS_OUTPUT.PUT_LINE('Invoking the procedure p2_noexcep ...');
 p2_noexcep(p_lid);
```


Copyright © Oracle Corporation, 2001. All rights reserved.

Removing Procedures

Drop a procedure stored in the database.

Syntax:

```
DROP PROCEDURE procedure_name
```

Example:

```
DROP PROCEDURE raise_salary;
```

```
Procedure dropped.
```


Copyright © Oracle Corporation, 2001. All rights reserved.

Summary

In this lesson, you should have learned that:

- A procedure is a subprogram that performs an action.
- You create procedures by using the `CREATE PROCEDURE` command.
- You can compile and save a procedure in the database.
- Parameters are used to pass data from the calling environment to the procedure.
- There are three parameter modes: `IN`, `OUT`, and `IN OUT`.

Copyright © Oracle Corporation, 2001. All rights reserved.

Summary

- Local subprograms are programs that are defined within the declaration section of another program.
- Procedures can be invoked from any tool or language that supports PL/SQL.
- You should be aware of the effect of handled and unhandled exceptions on transactions and calling procedures.
- You can remove procedures from the database by using the `DROP PROCEDURE` command.
- Procedures can serve as building blocks for an application.

Copyright © Oracle Corporation, 2001. All rights reserved.