Database Programming with PL/SQL

Managing Procedures and Functions

Objectives

This lesson covers the following objectives:

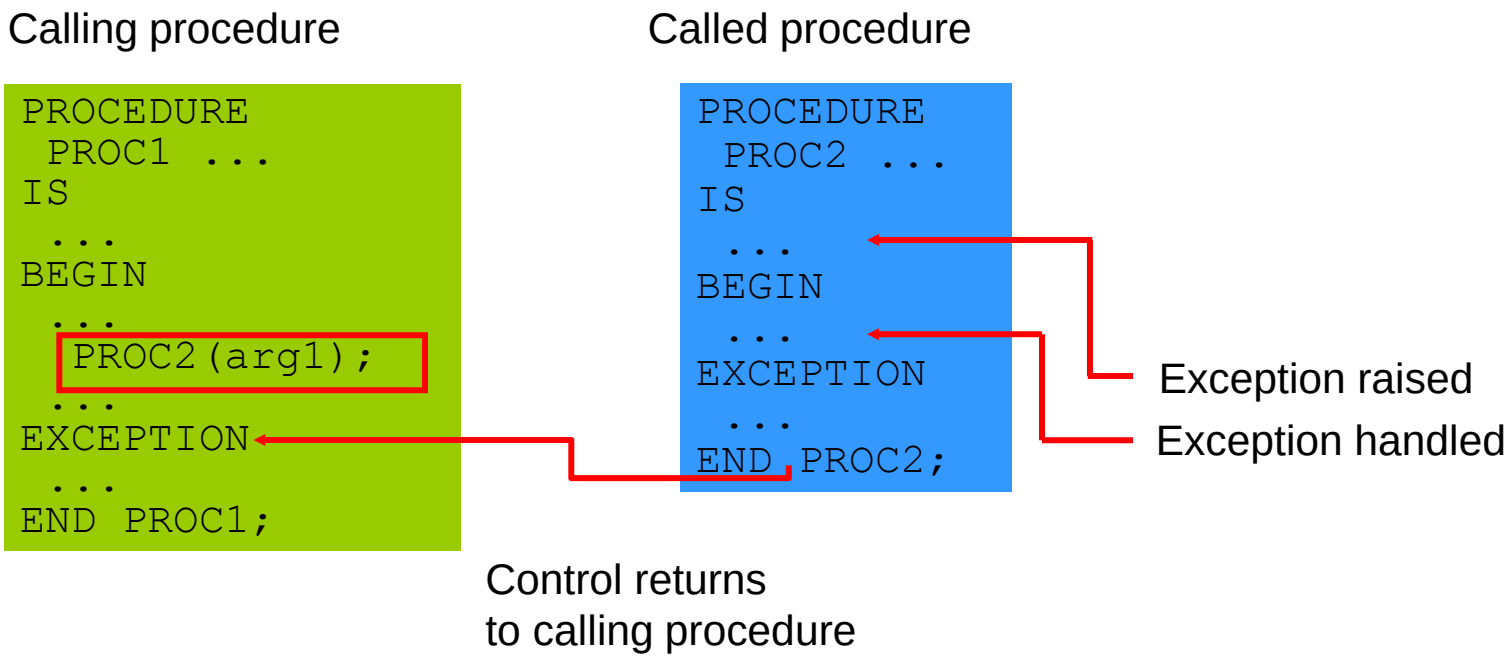
- Describe how exceptions are propagated
- Remove a function and a procedure
- Use Data Dictionary views to identify and manage stored programs

Purpose

In this lesson, you learn to manage procedures and functions.

To make your programs robust, you should always manage exception conditions by using the exception-handling features of PL/SQL.

Handled Exceptions



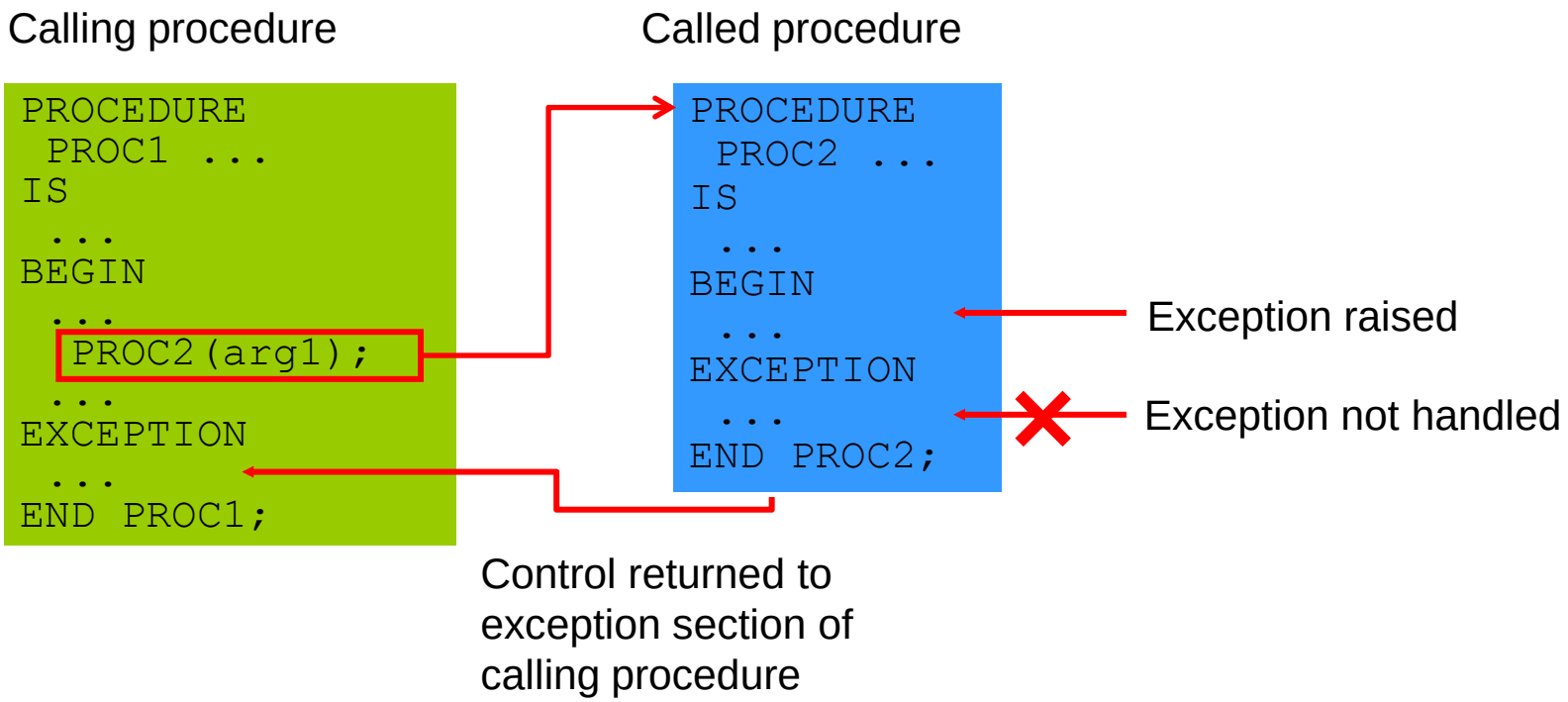
The following slides use procedures as examples, but the same rules apply to functions.

Handled Exceptions: Example

```
CREATE OR REPLACE PROCEDURE add_department(
  p_name VARCHAR2, p_mgr NUMBER, p_loc NUMBER) IS
BEGIN
  INSERT INTO DEPARTMENTS (department_id,
    department_name, manager_id, location_id)
  VALUES (DEPARTMENTS_SEQ.NEXTVAL, p_name, p_mgr, p_loc);
  DBMS_OUTPUT.PUT_LINE('Added Dept: '||p_name);
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error adding dept: '||p_name);
END;
```

```
BEGIN
  add_department('Media', 100, 1800);
  add_department('Editing', 99, 1800);
  add_department('Advertising', 101, 1800);
END;
```

Exceptions Not Handled



Exceptions Not Handled: Example

```
CREATE OR REPLACE PROCEDURE add_department_noex(
    p_name VARCHAR2, p_mgr NUMBER, p_loc NUMBER) IS
BEGIN
  INSERT INTO DEPARTMENTS (department_id,
    department_name, manager_id, location_id)
  VALUES (DEPARTMENTS_SEQ.NEXTVAL, p_name, p_mgr, p_loc);
  DBMS_OUTPUT.PUT_LINE('Added Dept: '||p_name);
END;
```

```
BEGIN
  add_department_noex('Media', 100, 1800);
  add_department_noex('Editing', 99, 1800);
  add_department_noex('Advertising', 101, 1800);
END;
```

Removing Procedures and Functions

You can remove a procedure or function that is stored in the database.

Syntax:

```
DROP {PROCEDURE procedure_name| FUNCTION function_name}
```

Examples:

```
DROP PROCEDURE raise_salary;
```

```
DROP FUNCTION get_sal;
```

Viewing Subprogram Names in the `USER_OBJECTS` Table

This example lists the names of all the PL/SQL functions that you own:

```
SELECT object_name
  FROM   USER_OBJECTS
  WHERE  object_type = 'FUNCTION';
```

OBJECT_NAME
TAX
DML_CALL_SQL

Viewing PL/SQL Source Code in the USER_SOURCE Table

This example shows the source code of the TAX function, which you own. Make sure you include ORDER BY line to see the lines of code in the correct sequence.

```
SELECT text
  FROM   USER_SOURCE
  WHERE  type = 'FUNCTION' AND name = 'TAX'
  ORDER BY line;
```

TEXT
FUNCTION tax(value IN NUMBER)
RETURN NUMBER IS
BEGIN
RETURN (value*0.08);
END tax;

Viewing Object Names and Source Code in Application Express

You can easily view subprogram information in Application Express:

- From SQL Workshop, click Object Browser, then Browse, and choose either Procedures or Functions as required. A list of subprograms appears.
- Click the required subprogram name. The source code of the subprogram appears.
- From here, you can edit and recompile it, or drop it if you want.

Terminology

Key terms used in this lesson included:

- `ALL_SOURCE`
- `USER_OBJECTS`
- `USER_SOURCE`

Summary

In this lesson, you should have learned how to:

- Describe how exceptions are propagated
- Remove a function and a procedure
- Use Data Dictionary views to identify and manage stored programs