Creating Procedures

Objectives

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

- Describe the uses of procedures
- Create client-side and server-side procedures
- Create procedures with parameters
- Declare subprograms
- Invoke a procedure
- Remove a procedure

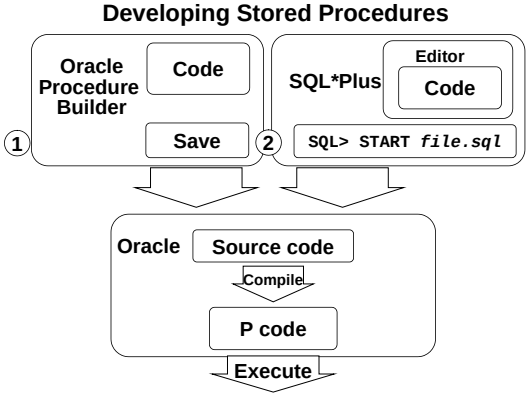
Overview of Procedures

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

Syntax for Creating Procedures

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

Developing Stored Procedures



Creating a Stored Procedure Using SQL*Plus

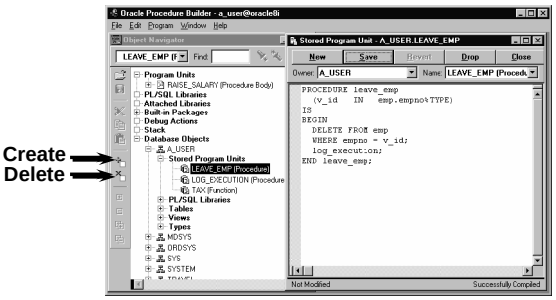
1. Enter the text of the CREATE PROCEDURE statement in an editor.
2. Run the script file to store the source code and compile the procedure.
3. Use SHOW ERRORS to see compilation errors.
4. When successfully compiled, the procedure is ready for execution.

Creating a Procedure Using Procedure Builder

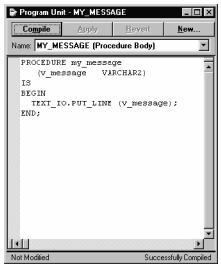
Procedure Builder allows you to:

- **Create a server-side procedure**
- **Create a client-side procedure**
- **Drag and drop procedures between client and server**

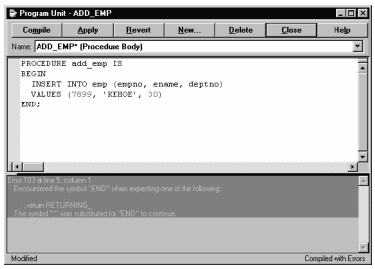
Creating Server-Side Procedures Using Procedure Builder



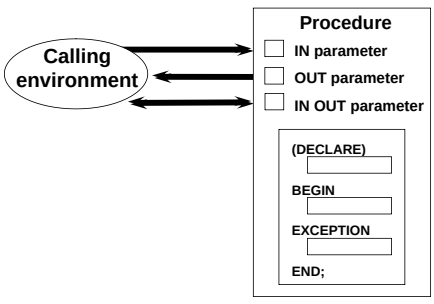
Creating Client-Side Procedures Using Procedure Builder



Navigating Compilation Errors in Procedure Builder



Procedural Parameter Modes



Creating Procedures with Parameters

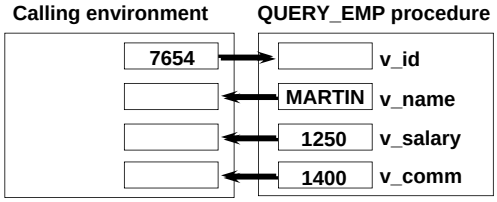
IN	OUT	IN OUT
Default	Must be specified	Must be specified
Value is passed into subprogram	Returned to calling environment	Passed into subprogram; returned to calling environment
Formal parameter acts as a constant	Uninitialized variable	Initialized variable
Actual parameter can be a literal, expression, 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_sal        NUMBER
SQL> VARIABLE g_commNUMBER
```

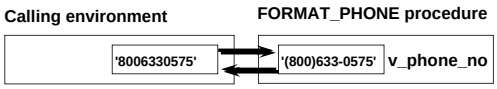
```
SQL> EXECUTE query_emp(7654,:g_name,:g_sal,:g_comm)
PL/SQL procedure successfully completed.
```

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

OUT Parameters and Procedure Builder

```
PL/SQL> .CREATE CHAR g_name LENGTH 10
PL/SQL> .CREATE NUMBER g_sal PRECISION 4
PL/SQL> .CREATE NUMBER g_comm PRECISION 4
PL/SQL> QUERY_EMP (7654, :g_name, :g_sal,
     +> :g_comm);
PL/SQL> TEXT_IO.PUT_LINE (:g_name || ' earns ' ||
     +> TO_CHAR(:g_sal) || ' and a commission of '
     +> || TO_CHAR(:g_comm));
MARTIN earns 1250 and a commission of 1400
```

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> PRINT g_phone_no
G_PHONE_NO
---------------
8006330575
```

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

SQL> PRINT g_phone_no
G_PHONE_NO
---------------
(800)633-0575
```

Invoking FORMAT_PHONE from Procedure Builder

```
PL/SQL> .CREATE CHAR g_phone_no LENGTH 15
PL/SQL> BEGIN
     +> :g_phone_no := '8006330575';
     +> END;
PL/SQL> FORMAT_PHONE (:g_phone_no);
PL/SQL> TEXT_IO.PUT_LINE (:g_phone_no);
(800)633-0575
```

Methods for Passing Parameters

- Positional
- Named
- Combination

DEFAULT Option for Parameters

```
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
```

Declaring Subprograms

```
CREATE OR REPLACE PROCEDURE LEAVE_EMP2
  (v_id   IN   emp.empno%TYPE)
IS
  PROCEDURE log_exec
  IS
  BEGIN
    INSERT INTO log_table (user_id, log_date)
    VALUES (user,sysdate);
  END log_exec;
BEGIN
  DELETE FROM emp
  WHERE empno = v_id;
  log_exec;
END leave_emp2;
```

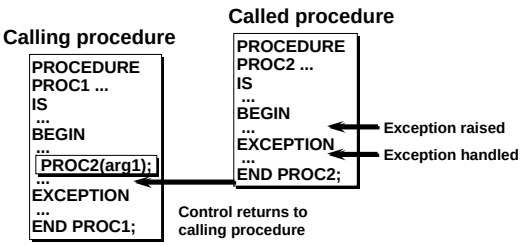
Invoking a Procedure from an Anonymous PL/SQL Block

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

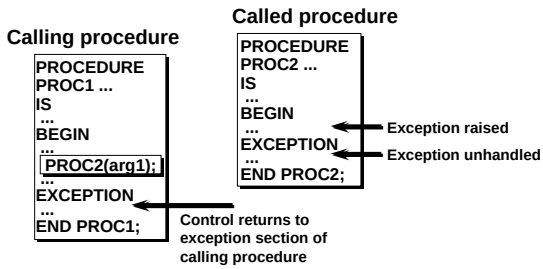
Invoking a Procedure from a Stored Procedure

```
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
  8    LOOP
  9      raise_salary(emp_rec.empno);  --invoke procedure
 10    END LOOP;
 11    COMMIT;
 12  END process_emps;
 13 /
```

Handled Exceptions



Unhandled Exceptions



Removing Procedures

- Using SQL*Plus:
  Drop a server-side procedure
- Using Procedure Builder:
  - Drop a server-side procedure
  - Delete a client-side procedure

Removing Server-Side Procedures

Using SQL*Plus:

Syntax

```
DROP PROCEDURE procedure_name
```

Example

```
SQL> DROP PROCEDURE raise_salary;
Procedure dropped.
```

Removing Server-Side Procedures

Using Procedure Builder:

1. **Connect to the database.**
2. **Expand the Database Objects node.**
3. **Expand the schema of the owner of the procedure.**
4. **Expand the Stored Program Units node.**
5. **Click the procedure you want to drop.**
6. **Click Delete in the Object Navigator.**
7. **Click Yes to confirm.**

Removing Client-Side Procedures

Using Procedure Builder:

1. **Expand the Program Units node.**
2. **Click the procedure you want to remove.**
3. **Click Delete in the Object Navigator.**
4. **Click Yes to confirm.**