Practices for Lesson 2: Creating Procedures

Chapter 2

Practices for Lesson 2: Overview

Overview

In this practice, you create, compile, and invoke procedures that issue DML and query commands. You also learn how to handle exceptions in procedures.

Note:

1. Before starting this practice, execute the `/home/oracle/labs/plpu/code_ex/cleanup_scripts/cleanup_02.sql` script.
2. If you missed a step in a practice, please run the appropriate solution script for that practice step before proceeding to the next step or the next practice.

Practice 2-1: Creating, Compiling, and Calling Procedures

Overview

In this practice, you create and invoke the `ADD_JOB` procedure and review the results. You also create and invoke a procedure called `UPD_JOB` to modify a job in the `JOBS` table and create and invoke a procedure called `DEL_JOB` to delete a job from the `JOBS` table. Finally, you create a procedure called `GET_EMPLOYEE` to query the `EMPLOYEES` table, retrieving the salary and job ID for an employee when provided with the employee ID.

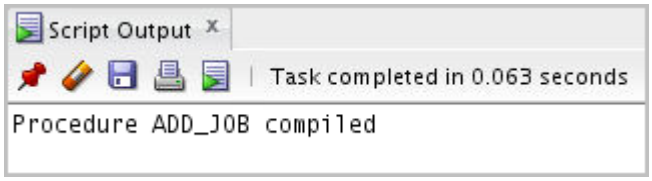
Note: Execute `cleanup_02.sql` from `/home/oracle/labs/plpu/code_ex/cleanup_scripts/` before performing the following task.

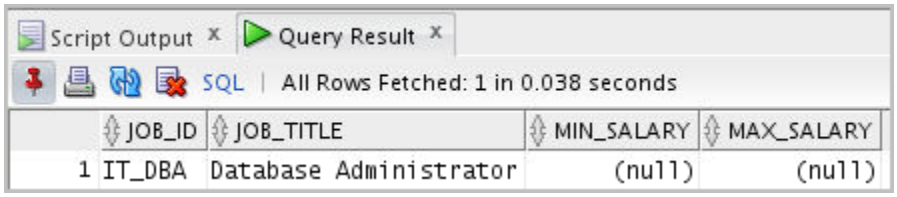
Task

1. Create, compile, and invoke the `ADD_JOB` procedure and review the results.
   a. Create a procedure called `ADD_JOB` to insert a new job into the `JOBS` table. Provide the ID and job title using two parameters.

      Note: You can create the procedure (and other objects) by entering the code in the SQL Worksheet area, and then click the Run Script (F5) icon. This creates and compiles the procedure. To find out whether or not the procedure has any errors, click the procedure name in the procedure node, and then select Compile from the pop-up menu.

   b. Invoke the procedure with `IT_DBA` as the job ID and `Database Administrator` as the job title. Query the `JOBS` table and view the results.





   c. Invoke your procedure again, passing a job ID of `ST_MAN` and a job title of `Stock Manager`. What happens and why?

2. Create a procedure called `UPD_JOB` to modify a job in the `JOBS` table.
   a. Create a procedure called `UPD_JOB` to update the job title. Provide the job ID and a new title using two parameters. Include the necessary exception handling if no update occurs.
   b. Invoke the procedure to change the job title of the job ID `IT_DBA` to `Data Administrator`. Query the `JOBS` table and view the results.

```
Script Output
Task completed in 0.154 seconds
Procedure UPD_JOB compiled
```

```
Script Output   Query Result
SQL | All Rows Fetched: 1 in 0.005 seconds
```

	JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
1	IT_DBA	Data Administrator	(null)	(null)

c. Test the exception-handling section of the procedure by trying to update a job that does not exist. You can use the job ID `IT_WEB` and the job title `Web Master`.

```
Script Output
Task completed in 0.014 seconds
Error starting at line : 61 in command -
EXECUTE upd_job ('IT_WEB', 'Web Master')
Error report -
ORA-20202: No job updated.
ORA-06512: at "ORA61.UPD_JOB", line 9
ORA-06512: at line 1
```

3. Create a procedure called `DEL_JOB` to delete a job from the `JOBS` table.

a. Create a procedure called `DEL_JOB` to delete a job. Include the necessary exception-handling code if no job is deleted.

```
Script Output
Task completed in 0.036 seconds
Procedure DEL_JOB compiled
```

b. Invoke the procedure using the job ID `IT_DBA`. Query the `JOBS` table and view the results.

```
Script Output   Query Result
SQL | All Rows Fetched: 0 in 0.001 seconds
```

	JOB_ID	JOB_TITLE	MIN_SAL...	MAX_SAL...

c. Test the exception-handling section of the procedure by trying to delete a job that does not exist. Use `IT_WEB` as the job ID. You should get the message that you included in the exception-handling section of the procedure as the output.

```
Script Output
Task completed in 0.017 seconds
Error starting at line : 94 in command -
EXECUTE del_job ('IT_WEB')
Error report -
ORA-20203: No jobs deleted.
ORA-06512: at "ORA61.DEL_JOB", line 6
ORA-06512: at line 1
```

4. Create a procedure called `GET_EMPLOYEE` to query the `EMPLOYEES` table, retrieving the salary and job ID for an employee when provided with the employee ID.
   a. Create a procedure that returns a value from the `SALARY` and `JOB_ID` columns for a specified employee ID. Remove syntax errors, if any, and then recompile the code.
   b. Execute the procedure using host variables for the two `OUT` parameters—one for the salary and the other for the job ID. Display the salary and job ID for employee ID 120.

```
PL/SQL procedure successfully completed.


  V_SALARY
----------
      8000
V_JOB
-----
ST_MAN
```

   c. Invoke the procedure again, passing an `EMPLOYEE_ID` of `300`. What happens and why?

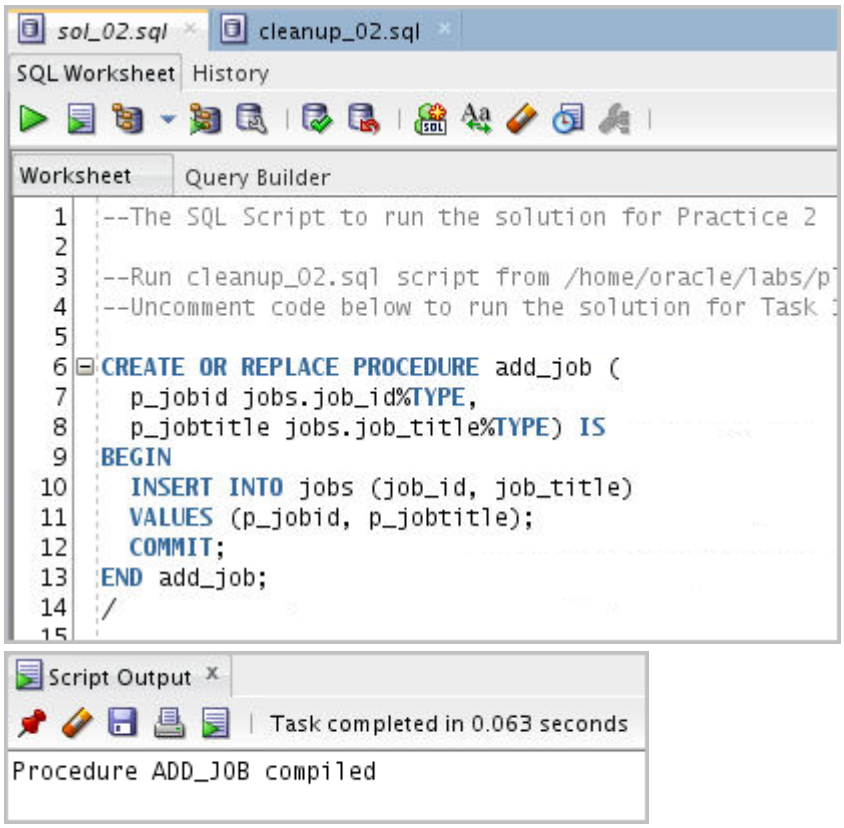
Solution 2-1: Creating, Compiling, and Calling Procedures

In this practice, you create and invoke the `ADD_JOB` procedure and review the results. You also create and invoke a procedure called `UPD_JOB` to modify a job in the `JOBS` table and create and invoke a procedure called `DEL_JOB` to delete a job from the `JOBS` table. Finally, you create a procedure called `GET_EMPLOYEE` to query the `EMPLOYEES` table, retrieving the salary and job ID for an employee when provided with the employee ID.

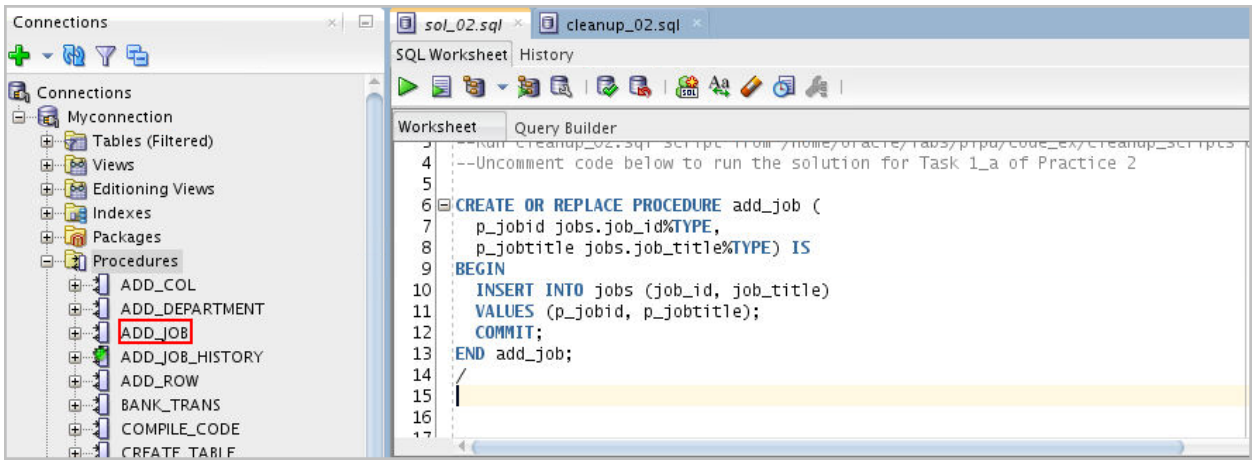
1. Create, compile, and invoke the `ADD_JOB` procedure and review the results.
   a. Create a procedure called `ADD_JOB` to insert a new job into the `JOBS` table. Provide the ID and job title using two parameters.

      Note: You can create the procedure (and other objects) by entering the code in the SQL Worksheet area, and then click the Run Script icon (or press F5). This creates and compiles the procedure. If the procedure generates an error message when you create it, click the procedure name in the procedure node, edit the procedure, and then select Compile from the pop-up menu.

      Open the `sol_02.sql` file in the `/home/oracle/labs/plpu/solns` directory. Uncomment and select the code for task 1_a. Click the Run Script icon (or press F5) on the SQL Worksheet toolbar to create and compile the procedure. The code and the result are displayed as follows:



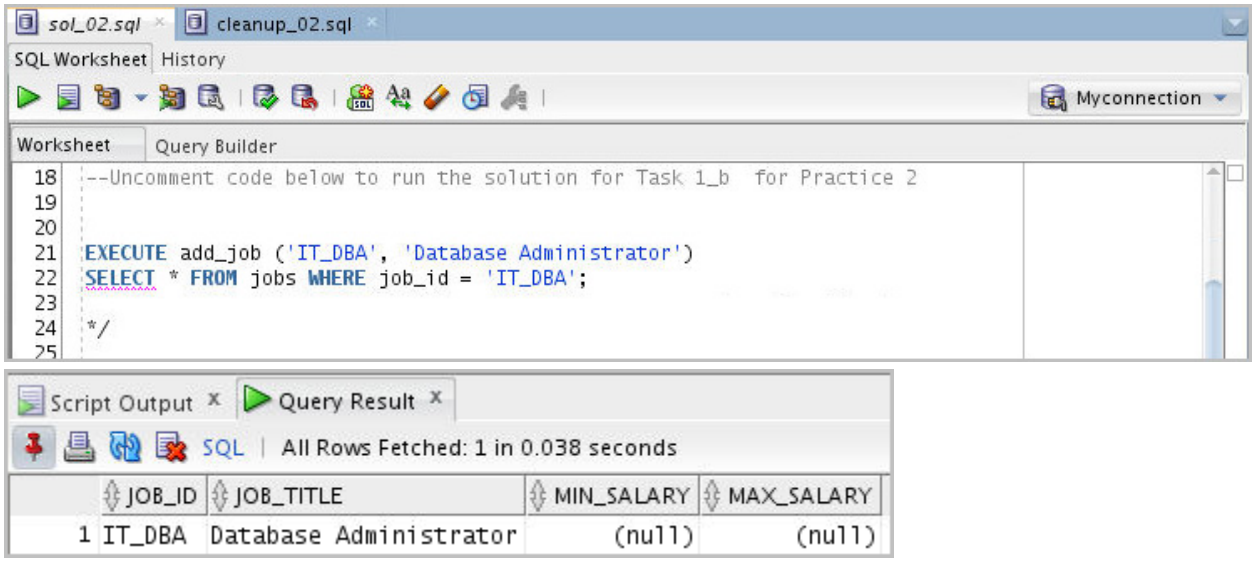
      To view the newly created procedure, click the Procedures node in the Object Navigator. If the newly created procedure is not displayed, right-click the Procedures node, and then select Refresh from the shortcut menu. The new procedure is displayed as follows:



b. Invoke the procedure with `IT_DBA` as the job ID and `Database Administrator` as the job title. Query the `JOBS` table and view the results.

Execute the code for Task 1_b from `sol_02.sql` script. The code and the result are displayed as follows:

Note: Be sure to comment the previous code before uncommenting the next set of code.



c. Invoke your procedure again, passing a job ID of `ST_MAN` and a job title of `Stock Manager`. What happens and why?

Run the code for Task 1_c from `sol_02.sql` script. The code and the result are displayed as follows:

An exception occurs because there is a Unique key integrity constraint on the `JOB_ID` column.

```
25
26  --Uncomment code below to run the solution for Task 1_c  for Practice 2
27
28  EXECUTE add_job ('ST_MAN', 'Stock Manager')
29
```

```
Error starting at line : 28 in command -
EXECUTE add_job ('ST_MAN', 'Stock Manager')
Error report -
ORA-00001: unique constraint (ORA61.JOB_ID_PK) violated
ORA-06512: at "ORA61.ADD_JOB", line 5
ORA-06512: at line 1
00001. 00000 -  "unique constraint (%s.%s) violated"
*Cause:    An UPDATE or INSERT statement attempted to insert a duplicate key.
           For Trusted Oracle configured in DBMS MAC mode, you may see
           this message if a duplicate entry exists at a different level.
*Action:   Either remove the unique restriction or do not insert the key.
```

2. Create a procedure called `UPD_JOB` to modify a job in the `JOBS` table.
   a. Create a procedure called `UPD_JOB` to update the job title. Provide the job ID and a new title by using two parameters. Include the necessary exception handling if no update occurs.

      Run the code for Task 2_a from the `sol_02.sql` script. The code and the result are displayed as follows:

```
30
31  --Uncomment code below to run the solution for Task 2_a  for Practice 2
32
33
34  CREATE OR REPLACE PROCEDURE upd_job (
35    p_jobid IN jobs.job_id%TYPE,
36    p_jobtitle IN jobs.job_title%TYPE) IS
37  BEGIN
38    UPDATE jobs
39    SET    job_title = p_jobtitle
40    WHERE  job_id = p_jobid;
41    IF SQL%NOTFOUND THEN
42      RAISE_APPLICATION_ERROR(-20202, 'No job updated.');
43    END IF;
44  END upd_job;
45  /
```

```
Script Output
Task completed in 0.154 seconds
Procedure UPD_JOB compiled
```

b. Invoke the procedure to change the job title of the job ID `IT_DBA` to `Data Administrator`. Query the `JOBS` table and view the results.

Run the code for Task 2_b from `sol_02.sql` script. The code and the result are displayed as follows:

```
sol_02.sql   cleanup_02.sql
SQL Worksheet   History
Myconnection
Worksheet   Query Builder
48
49  --Uncomment code below to run the solution for Task 2_b for Practice 2
50
51
52  EXECUTE upd_job ('IT_DBA', 'Data Administrator')
53  SELECT * FROM jobs WHERE job_id = 'IT_DBA';
54
```

```
Script Output   Query Result
SQL   All Rows Fetched: 1 in 0.005 seconds
```

	JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
1	IT_DBA	Data Administrator	(null)	(null)

c. Test the exception-handling section of the procedure by trying to update a job that does not exist. You can use the job ID `IT_WEB` and the job title `Web Master`.

Run the code for Task 2_c from `sol_02.sql` script. The code and the result are displayed as follows:

```
sol_02.sql   cleanup_02.sql
SQL Worksheet   History
Myconnection
Worksheet   Query Builder
54
55  */
56
57  --Uncomment code below to run the solution for Task 2_c for Practice 2
58
59
60
61  EXECUTE upd_job ('IT_WEB', 'Web Master')
62  SELECT * FROM jobs WHERE job_id = 'IT_WEB';
63
64  */
```

```
Script Output
Task completed in 0.014 seconds
Error starting at line : 61 in command -
EXECUTE upd_job ('IT_WEB', 'Web Master')
Error report -
ORA-20202: No job updated.
ORA-06512: at "ORA61.UPD_JOB", line 9
ORA-06512: at line 1
```

```
Script Output | Query Result
SQL | All Rows Fetched: 0 in 0.002 seconds
JOB_ID | JOB_TITLE | MIN_SAL... | MAX_SAL...
```

3. Create a procedure called `DEL_JOB` to delete a job from the `JOBS` table.
   a. Create a procedure called `DEL_JOB` to delete a job. Include the necessary exception-handling code if no job is deleted.

      Run the code for Task 3_a from `sol_02.sql` script. The code and the result are displayed as follows:

```
sol_02.sql | cleanup_02.sql
SQL Worksheet | History
Myconnection
Worksheet | Query Builder
65
66  --Uncomment code below to run the solution for Task 3_a for Practice 2
67
68
69
70  CREATE OR REPLACE PROCEDURE del_job (p_jobid jobs.job_id%TYPE) IS
71  BEGIN
72    DELETE FROM jobs
73    WHERE  job_id = p_jobid;
74    IF SQL%NOTFOUND THEN
75      RAISE_APPLICATION_ERROR(-20203, 'No jobs deleted.');
76    END IF;
77  END DEL_JOB;
78  /
```

```
Script Output
Task completed in 0.036 seconds
Procedure DEL_JOB compiled
```

   b. **To invoke the procedure and then query the JOBS table, uncomment and select the code under task 3_b in the `/home/oracle/labs/plpu/solns/sol_02.sql` script. Click the Run Script icon (or press F9) icon on the SQL Worksheet toolbar to invoke the procedure. Click the Query Result tab to see the code and the result displayed as follows:**

```
--Uncomment code below to run the solution for Task 3_b for Practice 2



EXECUTE del_job ('IT_DBA')
SELECT * FROM jobs WHERE job_id = 'IT_DBA';
```

Script Output | Query Result

All Rows Fetched: 0 in 0.001 seconds

JOB_ID	JOB_TITLE	MIN_SAL...	MAX_SAL...

c. Test the exception-handling section of the procedure by trying to delete a job that does not exist. Use `IT_WEB` as the job ID. You should get the message that you included in the exception-handling section of the procedure as the output.

To invoke the procedure and then query the JOBS table, uncomment and select the code under task 3_c in the `/home/oracle/labs/plpu/solns/sol_02.sql` script. Click the Run Script (F5) icon on the SQL Worksheet toolbar to invoke the procedure. The code and the result are displayed as follows:

```
--Uncomment code below to run the solution for Task 3_c for Practice 2

EXECUTE del_job ('IT_WEB')
```

Script Output

Task completed in 0.028 seconds

```
Error starting at line : 93 in command -
EXECUTE del_job ('IT_WEB')
Error report -
ORA-20203: No jobs deleted.
ORA-06512: at "ORA61.DEL_JOB", line 6
ORA-06512: at line 1
```

4. Create a procedure called `GET_EMPLOYEE` to query the `EMPLOYEES` table, retrieving the salary and job ID for an employee when provided with the employee ID.
   a. Create a procedure that returns a value from the `SALARY` and `JOB_ID` columns for a specified employee ID. Remove syntax errors, if any, and then recompile the code.

Uncomment and select the code for Task 4_a from the `sol_02.sql` script. Click the Run Script icon (or press F5) on the SQL Worksheet toolbar to create and compile the procedure. The code and the result are displayed as follows:

```
--Uncomment code below to run the solution for Task 4_a  for Practice 2



CREATE OR REPLACE PROCEDURE get_employee
    (p_empid IN  employees.employee_id%TYPE,
     p_sal   OUT employees.salary%TYPE,
     p_job   OUT employees.job_id%TYPE) IS
BEGIN
  SELECT  salary, job_id
  INTO    p_sal, p_job
  FROM    employees
  WHERE   employee_id = p_empid;
END get_employee;
/
```

```
Procedure GET_EMPLOYEE compiled
```

Note: If the newly created procedure is not displayed in the Object Navigator, right-click the Procedures node in the Object Navigator, and then select Refresh from the shortcut menu. Right-click the procedure's name in the Object Navigator, and then select Compile from the shortcut menu. The procedure is compiled.

b. Execute the procedure using host variables for the two `OUT` parameters—one for the salary and the other for the job ID. Display the salary and job ID for employee ID 120.

Uncomment and select the code under Task 4_b from `sol_02.sql` script. Click the Run Script icon (or press F5) on the SQL Worksheet toolbar to invoke the procedure. The code and the result are displayed as follows:

```
--Uncomment code below to run the solution for Task 4_b  for Practice 2



VARIABLE v_salary NUMBER
VARIABLE v_job    VARCHAR2(15)
EXECUTE get_employee(120, :v_salary, :v_job)
PRINT v_salary v_job
*/
```

```
PL/SQL procedure successfully completed.


  V_SALARY
----------
      8000
V_JOB
-----
ST_MAN
```

c. Invoke the procedure again, passing an `EMPLOYEE_ID` of `300`. What happens and why?

Uncomment and select the code under Task 4_c from `sol_02.sql` script. Click the Run Script icon (or press F5) on the SQL Worksheet toolbar to invoke the procedure. The code and the result are displayed as follows:

There is no employee in the `EMPLOYEES` table with an `EMPLOYEE_ID` of `300`. The `SELECT` statement retrieved no data from the database, resulting in a fatal PL/SQL error: `NO_DATA_FOUND` as follows: