Practices for Lesson 3: Creating Functions

Chapter 3

Practices for Lesson 3: Overview

Overview

In practice 3-1, you create, compile, and use the following:

- A function called `GET_JOB` to return a job title
- A function called `GET_ANNUAL_COMP` to return the annual salary computed from an employee's monthly salary and commission passed as parameters
- A procedure called `ADD_EMPLOYEE` to insert a new employee into the `EMPLOYEES` table

In practice 3-2, you are introduced to the basic functionality of the SQL Developer debugger:

- Create a procedure and a function.
- Insert breakpoints in the newly created procedure.
- Compile the procedure and function for debug mode.
- Debug the procedure and step into the code.
- Display and modify the subprograms' variables.

Note:

1. Before starting this practice, execute `/home/oracle/labs/plpu/code_ex/cleanup_scripts/cleanup_03.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 3-1: Creating Functions

Overview

In this practice, you create, compile, and use stored functions and a procedure.

Note: Execute `cleanup_03.sql` from `/home/oracle/labs/plpu/code_ex/cleanup_scripts/` before performing the following tasks.

Task

1. Create and invoke the `GET_JOB` function to return a job title.
   a. Create and compile a function called `GET_JOB` to return a job title.

      Create a `VARCHAR2` host variable called `b_title`, allowing a length of 35 characters. Invoke the function with job ID `SA_REP` to return the value in the host variable, and then print the host variable to view the result.

```
Script Output
Task completed in 0.019 seconds
PL/SQL procedure successfully completed.


B_TITLE
-------
Sales Representative
```

2. Create a function called `GET_ANNUAL_COMP` to return the annual salary computed from an employee's monthly salary and commission passed as parameters.
   a. Create the `GET_ANNUAL_COMP` function, which accepts parameter values for the monthly salary and commission. Either or both values passed can be `NULL`, but the function should still return a non-`NULL` annual salary. Use the following basic formula to calculate the annual salary:

      `(salary*12) + (commission_pct*salary*12)`

   b. Use the function in a `SELECT` statement against the `EMPLOYEES` table for employees in department 30.

Query Result

SQL | All Rows Fetched: 6 in 0.075 seconds

	EMPLOYEE_ID	LAST_NAME	Annual Compensation
1	114	Raphaely	132000
2	115	Khoo	37200
3	116	Baida	34800
4	117	Tobias	33600
5	118	Himuro	31200
6	119	Colmenares	30000

3. Create a procedure, `ADD_EMPLOYEE`, to insert a new employee into the `EMPLOYEES` table. The procedure should call a `VALID_DEPTID` function to check whether the department ID specified for the new employee exists in the `DEPARTMENTS` table.

a. Create a function called `VALID_DEPTID` to validate a specified department ID and return a `BOOLEAN` value of `TRUE` if the department exists.

b. Create the `ADD_EMPLOYEE` procedure to add an employee to the `EMPLOYEES` table. The row should be added to the `EMPLOYEES` table if the `VALID_DEPTID` function returns `TRUE`; otherwise, alert the user with an appropriate message. Provide the following parameters:
   - `first_name`
   - `last_name`
   - `email`
   - `job`: Use `'SA_REP'` as the default value.
   - `mgr`: Use 145 as the default value.
   - `sal`: Use 1000 as the default value.
   - `comm`: Use 0 as the default value.
   - `deptid`: Use 30 as the default value.
   - Use the `EMPLOYEES_SEQ` sequence to set the `employee_id` column.
   - Set the `hire_date` column to `TRUNC(SYSDATE)`.

c. Call `ADD_EMPLOYEE` for the name `'Jane Harris'` in department `15`, leaving other parameters with their default values. What is the result?

d. Add another employee named Joe Harris in department 80, leaving the remaining parameters with their default values. What is the result?

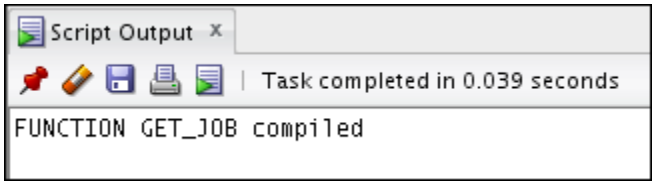
Solution 3-1: Creating Functions

In this practice, you create, compile, and use stored functions and a procedure.

1. Create and invoke the `GET_JOB` function to return a job title.
   a. Create and compile a function called `GET_JOB` to return a job title.

      Open the `/home/oracle/labs/plpu/solns/sol_03.sql` script. Uncomment and select the code under Task 1_a. Click the Run Script icon (or press F5) on the SQL Worksheet toolbar to create and compile the function. The code and the result are displayed as follows:

```
CREATE OR REPLACE FUNCTION get_job (p_jobid IN jobs.job_id%type)
 RETURN jobs.job_title%type IS
  v_title jobs.job_title%type;
BEGIN
  SELECT job_title
  INTO v_title
  FROM jobs
  WHERE job_id = p_jobid;
  RETURN v_title;
END get_job;
/
```



Note: If you encounter an "**access control list (ACL) error**" while executing this step, please perform the following workaround:

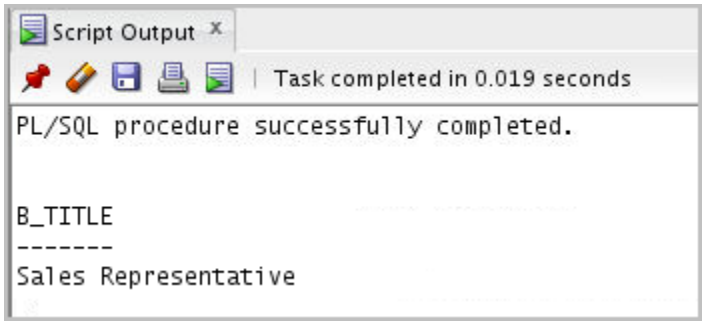
   a. Open SQL*Plus.
   b. Connect as SYSDBA.
   c. Execute the following code:

```
BEGIN
DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
host => '127.0.0.1',
ace => xs$ace_type(privilege_list => xs$name_list('jdwp'),
principal_name => 'ora61',
principal_type => xs_acl.ptype_db));
END;
/
```

   b. Create a `VARCHAR2` host variable called `b_title`, allowing a length of 35 characters. Invoke the function with job ID `SA_REP` to return the value in the host variable, and then print the host variable to view the result.

Uncomment and select the code under Task 1_b. Click the Run Script icon (or press F5) on the SQL Worksheet toolbar to create and compile the function. The code and the result are displayed as follows:

```
VARIABLE b_title VARCHAR2(35)
EXECUTE :b_title := get_job ('SA_REP');
PRINT b_title
```



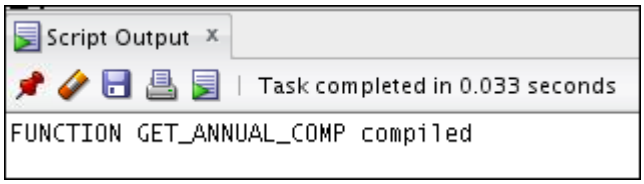
Note: Be sure to add the comments back to the previous code before executing the next set of code. Alternatively, you can select the complete code before using the Run Script icon (or press F5) to execute it.

2. Create a function called `GET_ANNUAL_COMP` to return the annual salary computed from an employee's monthly salary and commission passed as parameters.
   a. Create the `GET_ANNUAL_COMP` function, which accepts parameter values for the monthly salary and commission. Either or both values passed can be `NULL`, but the function should still return a non-`NULL` annual salary. Use the following basic formula to calculate the annual salary:

      `(salary*12) + (commission_pct*salary*12)`

      Uncomment and select the code under Task 2_a. Click the Run Script icon (or press F5) on the SQL Worksheet toolbar to create and compile the function. The code and the result are displayed as follows:

```
CREATE OR REPLACE FUNCTION get_annual_comp(
  p_sal  IN employees.salary%TYPE,
  p_comm IN employees.commission_pct%TYPE)
 RETURN NUMBER IS
BEGIN
  RETURN (NVL(p_sal,0) * 12 + (NVL(p_comm,0) * nvl(p_sal,0) *
12));
END get_annual_comp;
/
```



b. Use the function in a `SELECT` statement against the `EMPLOYEES` table for employees in department 30.

Uncomment and select the code under Task 2_b. Click the Run Script icon (or press F5) on the SQL Worksheet toolbar to create and compile the function. The code and the result are displayed as follows:

```
SELECT employee_id, last_name,
       get_annual_comp(salary,commission_pct) "Annual
Compensation"
FROM   employees
WHERE department_id=30
/
```

Query Result

SQL | All Rows Fetched: 6 in 0.075 seconds

	EMPLOYEE_ID	LAST_NAME	Annual Compensation
1	114	Raphaely	132000
2	115	Khoo	37200
3	116	Baida	34800
4	117	Tobias	33600
5	118	Himuro	31200
6	119	Colmenares	30000

3. Create a procedure, `ADD_EMPLOYEE`, to insert a new employee into the `EMPLOYEES` table. The procedure should call a `VALID_DEPTID` function to check whether the department ID specified for the new employee exists in the `DEPARTMENTS` table.

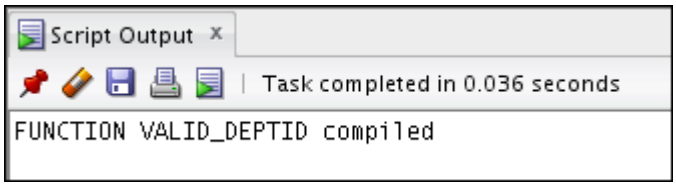
a. Create a function called `VALID_DEPTID` to validate a specified department ID and return a `BOOLEAN` value of `TRUE` if the department exists.

Uncomment and select the code under Task 3_a. Click the Run Script icon (or press F5) on the SQL Worksheet toolbar to create the function. The code and the result are displayed as follows:

```
CREATE OR REPLACE FUNCTION valid_deptid(
  p_deptid IN departments.department_id%TYPE)
  RETURN BOOLEAN IS
  v_dummy  PLS_INTEGER;

BEGIN
  SELECT  1
  INTO    v_dummy
  FROM    departments
  WHERE   department_id = p_deptid;
  RETURN  TRUE;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
```

```
    RETURN FALSE;
END valid_deptid;
/
```



b. Create the `ADD_EMPLOYEE` procedure to add an employee to the `EMPLOYEES` table. The row should be added to the `EMPLOYEES` table if the `VALID_DEPTID` function returns `TRUE`; otherwise, alert the user with an appropriate message. Provide the following parameters:
   - `first_name`
   - `last_name`
   - `email`
   - `job`: Use `'SA_REP'` as the default value.
   - `mgr`: Use 145 as the default value.
   - `sal`: Use 1000 as the default value.
   - `comm`: Use 0 as the default value.
   - `deptid`: Use 30 as the default value.
   - Use the `EMPLOYEES_SEQ` sequence to set the `employee_id` column.
   - Set the `hire_date` column to `TRUNC(SYSDATE)`.

Uncomment and select the code under Task 3_b. 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:

```
CREATE OR REPLACE PROCEDURE add_employee(
   p_first_name employees.first_name%TYPE,
   p_last_name  employees.last_name%TYPE,
   p_email      employees.email%TYPE,
   p_job        employees.job_id%TYPE        DEFAULT 'SA_REP',
   p_mgr        employees.manager_id%TYPE     DEFAULT 145,
   p_sal        employees.salary%TYPE         DEFAULT 1000,
   p_comm       employees.commission_pct%TYPE DEFAULT 0,
   p_deptid     employees.department_id%TYPE  DEFAULT 30) IS
BEGIN
 IF valid_deptid(p_deptid) THEN
   INSERT INTO employees(employee_id, first_name, last_name,
email,
     job_id, manager_id, hire_date, salary, commission_pct,
department_id)
```

```
   VALUES (employees_seq.NEXTVAL, p_first_name, p_last_name,
p_email,
     p_job, p_mgr, TRUNC(SYSDATE), p_sal, p_comm, p_deptid);
 ELSE
   RAISE_APPLICATION_ERROR (-20204, 'Invalid department ID. Try
again.');
 END IF;
END add_employee;
/
```

```
Script Output
Task completed in 0.037 seconds
PROCEDURE ADD_EMPLOYEE compiled
```

c. Call `ADD_EMPLOYEE` for the name `'Jane Harris'` in department 15, leaving other parameters with their default values. What is the result?

Uncomment and select the code under Task 3_c. 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:

```
EXECUTE add_employee('Jane','Harris','JAHARRIS',p_deptid=> 15)
```

```
Script Output
Task completed in 0.016 seconds
Error starting at line : 111 in command -
EXECUTE add_employee('Jane', 'Harris', 'JAHARRIS', p_deptid=> 15)
Error report -
ORA-20204: Invalid department ID. Try again.
ORA-06512: at "ORA61.ADD_EMPLOYEE", line 17
ORA-06512: at line 1
```

d. Add another employee named Joe Harris in department 80, leaving the remaining parameters with their default values. What is the result?

Uncomment and select the code under Task 3_d. 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:

```
EXECUTE add_employee('Joe', 'Harris', 'JOHARRIS', p_deptid=> 80)
```

```
Script Output
Task completed in 0.012 seconds
PL/SQL procedure successfully completed.
```