**Zhumagali Kanat IT2-2003**

**Practice 8**

* + 1. Create a package specification and body called JOB\_PKG, containing a copy of your ADD\_JOB, UPD\_JOB, and DEL\_JOB procedures as well as your GET\_JOB function.

**Note:** Use the code from your previously saved procedures and functions when creating the package. You can copy the code in a procedure or function, and then paste the code into the appropriate section of the package.

* + - 1. Create the package specification including the procedures and function headings as public constructs.
      2. Create the package body with the implementations for each of the subprograms.
         1. The ADD\_JOB, UPD\_JOB, and DEL\_JOB procedures
         2. The GET\_JOB function
      3. Invoke your ADD\_JOB package procedure by passing the values IT\_SYSAN and SYSTEMS ANALYST as parameters.
      4. Query the JOBS table to see the result.

create or replace package job\_pkg is

procedure add\_job (j\_id jobs.job\_id%type, j\_title in jobs.job\_title%type);

procedure upd\_job(

j\_id jobs.job\_id%type,

j\_ntitle jobs.job\_title%type);

procedure del\_job(

j\_id jobs.job\_id%type

);

function get\_job(j\_id jobs.job\_id%type) return jobs.job\_title%type;

end job\_pkg;

create or replace package body job\_pkg is

procedure add\_job(

j\_id in jobs.job\_id%type,

j\_title in jobs.job\_title%type) is

begin

insert into jobs(job\_id, job\_title) values (j\_id,j\_title);

end add\_job;

procedure upd\_job(

j\_id jobs.job\_id%type,

j\_ntitle jobs.job\_title%type) is

begin

update jobs set job\_title = j\_ntitle where job\_id = j\_id;

if(sql%rowcount = 0) then dbms\_output.put\_line ('No data found for updating...');

else dbms\_output.put\_line('Nice! ');

end if;

exception when others then dbms\_output.put\_line('error');

end upd\_job;

procedure del\_job(

j\_id jobs.job\_id%type

) is

begin

delete from jobs where job\_id = j\_id;

if(sql%rowcount = 0) then dbms\_output.put\_line ('No data found for deleting...');

else dbms\_output.put\_line('Nice! ');

end if;

exception

when others then dbms\_output.put\_line('error');

end del\_job;

function get\_job(j\_id jobs.job\_id%type) return jobs.job\_title%type is j\_title jobs.job\_title%type;

begin

select job\_title into j\_title from jobs where job\_id = j\_id;

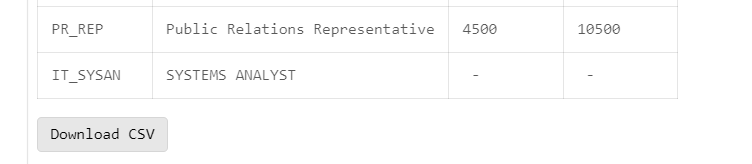
return j\_title;

end get\_job;

end job\_pkg;

execute job\_pkg.add\_job('IT\_SYSAN', 'SYSTEMS ANALYST')

select \* from jobs



2. Create and invoke a package that contains private and public constructs.

* + - 1. Create a package specification and a package body called EMP\_PKG that contains the following procedures and function that you created earlier:
         * ADD\_EMPLOYEE procedure as a *public* construct
         * GET\_EMPLOYEE procedure as a *public* construct
         * VALID\_DEPTID function as a *private* construct

1. Invoke the EMP\_PKG.ADD\_EMPLOYEE procedure, using department ID 15 for the employee Jane Harris with the email ID JAHARRIS. Because department ID 15 does not exist, you should get an error message as specified in the exception handler of your procedure.
2. Invoke the ADD\_EMPLOYEE package procedure by using department ID 80 for employee David Smith with the email ID DASMITH.
3. Query the EMPLOYEES table to verify that the new employee was added.

create or replace package emp\_pkg is

procedure add\_employee(

first\_name employees.first\_name%TYPE,

last\_name employees.last\_name%TYPE,

email employees.email%TYPE := 'mail',

job employees.job\_id%TYPE := 'SA\_REP',

mgr employees.manager\_id%TYPE := 145,

sal employees.salary%TYPE := 100,

comm employees.commission\_pct%TYPE := 0,

deptid employees.department\_id%TYPE := 30);

procedure get\_employee (

e\_id in employees.employee\_id%type,

j\_id out employees.job\_id%type,

salary out employees.salary%type);

end emp\_pkg;

create or replace package body emp\_pkg is

function valid\_deptid (department\_id departments.department\_id%type ) return boolean is

begin

if department\_id is null then return false;

else return true;

end if;

end valid\_deptid;

procedure add\_employee(

first\_name employees.first\_name%TYPE,

last\_name employees.last\_name%TYPE,

email employees.email%TYPE := 'mail',

job employees.job\_id%TYPE := 'SA\_REP',

mgr employees.manager\_id%TYPE := 145,

sal employees.salary%TYPE := 100,

comm employees.commission\_pct%TYPE := 0,

deptid employees.department\_id%TYPE := 30

)

is

begin

IF valid\_deptid(deptid) then

insert into employees(employee\_id, first\_name, last\_name, email, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id)

values (EMPLOYEES\_SEQ.nextval, first\_name, last\_name, email, TRUNC(SYSDATE), job, sal, comm, mgr, deptid);

DBMS\_OUTPUT.PUT\_LINE('TRUE');

else

DBMS\_OUTPUT.PUT\_LINE('FALSE');

end if;

end add\_employee;

procedure get\_employee (

e\_id in employees.employee\_id%type,

j\_id out employees.job\_id%type,

salary out employees.salary%type

) is

begin

select salary, job\_id into salary, j\_id from employees where employee\_id = e\_id;

exception

when no\_data\_found then dbms\_output.put\_line('No data found for this id');

when others then dbms\_output.put\_line('Error');

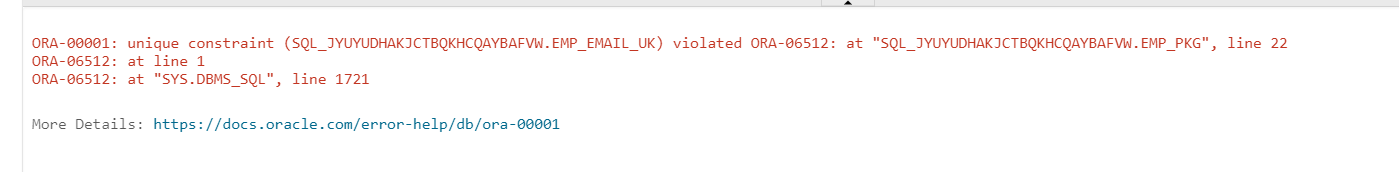
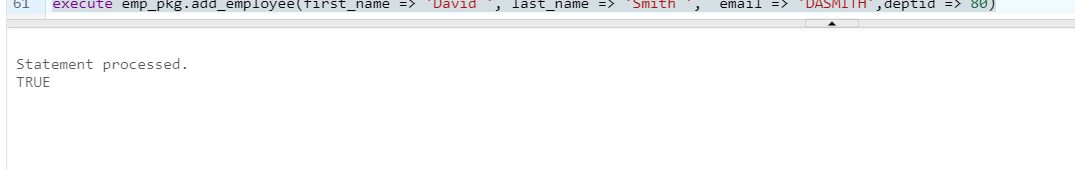
end get\_employee;

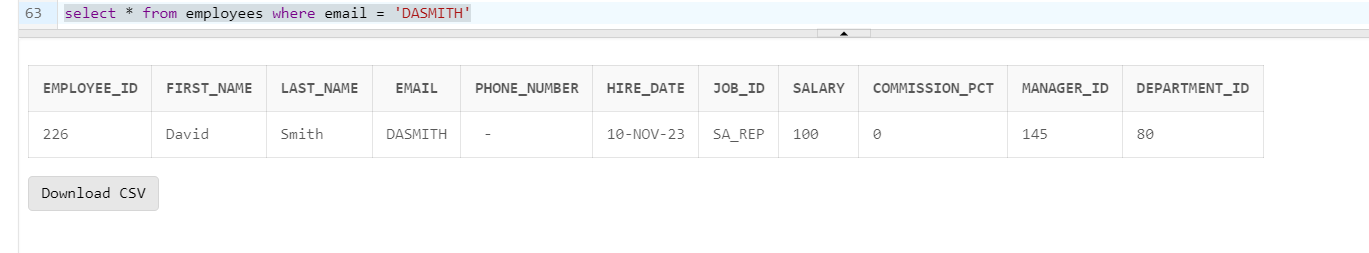
end emp\_pkg;

execute emp\_pkg.add\_employee(first\_name => 'Jane', last\_name => 'Harris', deptid => 15)

execute emp\_pkg.add\_employee(first\_name => 'David ', last\_name => 'Smith ', email => 'DASsMITH',deptid => 80)

select \* from employees where email = 'DASMITH'



3. Modify the code for the EMP\_PKG package, and overload the ADD\_EMPLOYEE procedure.

* + - 1. In the package specification, add a new procedure called ADD\_EMPLOYEE that accepts the following three parameters:
         1. First name
         2. Last name
         3. Department ID
      2. Click Run Script to create the package. Compile the package.
      3. Implement the new ADD\_EMPLOYEE procedure in the package body as follows:
         1. Format the email address in uppercase characters, using the first letter of the first name concatenated with the first seven letters of the last name.
         2. The procedure should call the existing ADD\_EMPLOYEE procedure to perform the actual INSERT operation using its parameters and formatted email to supply the values.
         3. Click Run Script to create the package. Compile the package.
      4. Invoke the new ADD\_EMPLOYEE procedure using the name Samuel Joplin to be added to department 30.
      5. Confirm that the new employee was added to the EMPLOYEES table.

create or replace package emp\_pkg is

procedure add\_employee(

first\_name employees.first\_name%TYPE,

last\_name employees.last\_name%TYPE,

email employees.email%TYPE := 'mail',

job employees.job\_id%TYPE := 'SA\_REP',

mgr employees.manager\_id%TYPE := 145,

sal employees.salary%TYPE := 100,

comm employees.commission\_pct%TYPE := 0,

deptid employees.department\_id%TYPE := 30);

procedure add\_employee(

first\_name employees.first\_name%TYPE,

last\_name employees.last\_name%TYPE,

deptid employees.department\_id%TYPE := 30);

procedure get\_employee (

e\_id in employees.employee\_id%type,

j\_id out employees.job\_id%type,

salary out employees.salary%type);

end emp\_pkg;

create or replace package body emp\_pkg is

function valid\_deptid (department\_id departments.department\_id%type ) return boolean is

begin

if department\_id is null then return false;

else return true;

end if;

end valid\_deptid;

procedure add\_employee(

first\_name employees.first\_name%TYPE,

last\_name employees.last\_name%TYPE,

email employees.email%TYPE := 'mail',

job employees.job\_id%TYPE := 'SA\_REP',

mgr employees.manager\_id%TYPE := 145,

sal employees.salary%TYPE := 100,

comm employees.commission\_pct%TYPE := 0,

deptid employees.department\_id%TYPE := 30

)

is

begin

IF valid\_deptid(deptid) then

insert into employees(employee\_id, first\_name, last\_name, email, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id)

values (EMPLOYEES\_SEQ.nextval, first\_name, last\_name, email, TRUNC(SYSDATE), job, sal, comm, mgr, deptid);

DBMS\_OUTPUT.PUT\_LINE('TRUE');

else

DBMS\_OUTPUT.PUT\_LINE('FALSE');

end if;

end add\_employee;

procedure add\_employee(

first\_name employees.first\_name%TYPE,

last\_name employees.last\_name%TYPE,

deptid employees.department\_id%TYPE := 30) is em employees.email%type;

begin

em:= UPPER(substr(first\_name, 1, 1) || substr(last\_name,1,7));

add\_employee(first\_name, last\_name, em);

end add\_employee;

procedure get\_employee (

e\_id in employees.employee\_id%type,

j\_id out employees.job\_id%type,

salary out employees.salary%type

) is

begin

select salary, job\_id into salary, j\_id from employees where employee\_id = e\_id;

exception

when no\_data\_found then dbms\_output.put\_line('No data found for this id');

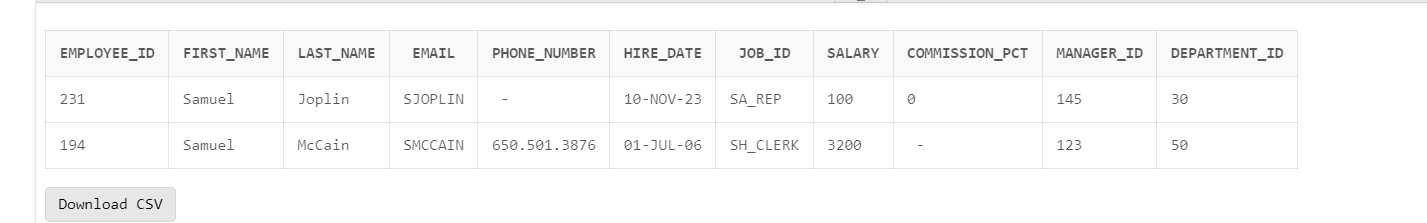
when others then dbms\_output.put\_line('Error');

end get\_employee;

end emp\_pkg;

execute emp\_pkg.add\_employee( 'Samuel', 'Joplin', 30)

select \* from employees where first\_name = 'Samuel'



4. In the EMP\_PKG package, create two overloaded functions called GET\_EMPLOYEE:

* + - 1. In the package specification, add the following functions:
         1. The GET\_EMPLOYEE function that accepts the parameter called p\_emp\_id based on the employees.employee\_id%TYPE type. This function should return an EMPLOYEES%ROWTYPE.
         2. The GET\_EMPLOYEE function that accepts the parameter called p\_family\_name of type employees.last\_name%TYPE. This function should return an EMPLOYEES%ROWTYPE.
      2. Click the Run Script (F5) icon to re-create the package. Compile the package.
      3. In the package body:
         1. Implement the first GET\_EMPLOYEE function to query an employee using the employee’s ID.
         2. Implement the second GET\_EMPLOYEE function to use the equality operator on the value supplied in the p\_ family\_name parameter.
      4. Click Run Script to re-create the package. Compile the package.
      5. Add a utility procedure PRINT\_EMPLOYEE to the EMP\_PKG package as follows:
         1. The procedure accepts an EMPLOYEES%ROWTYPE as a parameter.
         2. The procedure displays the following for an employee on one line, using the DBMS\_OUTPUT package:

- department\_id

- employee\_id

- first\_name

- last\_name

- job\_id

- salary

f. Click Run Script to re-create the package. Compile the package.

g. Use an anonymous block to invoke the EMP\_PKG.GET\_EMPLOYEE function with an employee ID of 100 and family name of 'Joplin'. Use the PRINT\_EMPLOYEE procedure to display the results for each row returned.

create or replace package emp\_pkg is

procedure add\_employee(

first\_name employees.first\_name%TYPE,

last\_name employees.last\_name%TYPE,

email employees.email%TYPE := 'mail',

job employees.job\_id%TYPE := 'SA\_REP',

mgr employees.manager\_id%TYPE := 145,

sal employees.salary%TYPE := 100,

comm employees.commission\_pct%TYPE := 0,

deptid employees.department\_id%TYPE := 30);

procedure add\_employee(

first\_name employees.first\_name%TYPE,

last\_name employees.last\_name%TYPE,

deptid employees.department\_id%TYPE := 30);

procedure get\_employee (

e\_id in employees.employee\_id%type,

j\_id out employees.job\_id%type,

salary out employees.salary%type);

function get\_employee(p\_emp\_id employees.employee\_id%type) return employees%rowtype;

function get\_employee(p\_family\_name employees.last\_name%type) return employees%rowtype;

procedure print\_employee(emp\_info employees%ROWTYPE);

end emp\_pkg;

create or replace package body emp\_pkg is

function get\_employee(p\_emp\_id employees.employee\_id%type) return employees%rowtype is emp\_info employees%rowtype;

begin

select \* into emp\_info from employees where employee\_id = p\_emp\_id;

return emp\_info;

end get\_employee;

function get\_employee(p\_family\_name employees.last\_name%type) return employees%rowtype is emp\_info employees%rowtype;

begin

select \* into emp\_info from employees where last\_name = p\_family\_name;

return emp\_info;

end get\_employee;

function valid\_deptid (department\_id departments.department\_id%type ) return boolean is

begin

if department\_id is null then return false;

else return true;

end if;

end valid\_deptid;

procedure add\_employee(

first\_name employees.first\_name%TYPE,

last\_name employees.last\_name%TYPE,

email employees.email%TYPE := 'mail',

job employees.job\_id%TYPE := 'SA\_REP',

mgr employees.manager\_id%TYPE := 145,

sal employees.salary%TYPE := 100,

comm employees.commission\_pct%TYPE := 0,

deptid employees.department\_id%TYPE := 30

)

is

begin

IF valid\_deptid(deptid) then

insert into employees(employee\_id, first\_name, last\_name, email, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id)

values (EMPLOYEES\_SEQ.nextval, first\_name, last\_name, email, TRUNC(SYSDATE), job, sal, comm, mgr, deptid);

DBMS\_OUTPUT.PUT\_LINE('TRUE');

else

DBMS\_OUTPUT.PUT\_LINE('FALSE');

end if;

end add\_employee;

procedure add\_employee(

first\_name employees.first\_name%TYPE,

last\_name employees.last\_name%TYPE,

deptid employees.department\_id%TYPE := 30) is em employees.email%type;

begin

em:= UPPER(substr(first\_name, 1, 1) || substr(last\_name,1,7));

add\_employee(first\_name, last\_name, em);

end add\_employee;

procedure get\_employee (

e\_id in employees.employee\_id%type,

j\_id out employees.job\_id%type,

salary out employees.salary%type

) is

begin

select salary, job\_id into salary, j\_id from employees where employee\_id = e\_id;

exception

when no\_data\_found then dbms\_output.put\_line('No data found for this id');

when others then dbms\_output.put\_line('Error');

end get\_employee;

procedure print\_employee(emp\_info employees%ROWTYPE) is

begin

dbms\_output.put\_line('Dep ID: ' || emp\_info.department\_id ||

' Emp ID: ' || emp\_info.employee\_id ||

' First Name: ' || emp\_info.first\_name ||

' Last Name: ' || emp\_info.last\_name ||

' Job ID: ' || emp\_info.job\_id ||

' Salary: ' || emp\_info.salary);

end print\_employee;

end emp\_pkg;

declare

e\_id employees%ROWTYPE;

e\_n employees%ROWTYPE;

begin

e\_id := emp\_pkg.get\_employee(100);

e\_n := emp\_pkg.get\_employee('Joplin');

emp\_pkg.print\_employee(e\_id);

emp\_pkg.print\_employee(e\_n);

end;

