**Zhumagali Kanat IT2-2003**

**Practice 4.1 (Composite Data Types)**

1. Within the PL/SQL block create a record to store information about yourself: name, year of birth, place of birth. Output the information to the screen.

DECLARE

TYPE my\_information is record (

last\_name varchar2(20),

year\_of\_birth number,

place\_of\_birth varchar2(20)

);

info my\_information;

BEGIN

info.last\_name := 'Zhumagali';

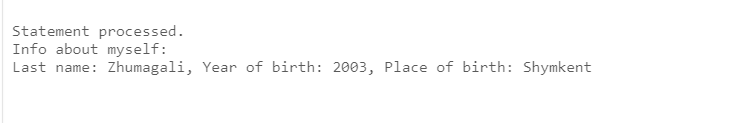
info.year\_of\_birth := 2003;

info.place\_of\_birth := 'Shymkent';

DBMS\_OUTPUT.PUT\_LINE('Info about myself: ');

DBMS\_OUTPUT.PUT\_LINE('Last name: ' ||info.last\_name || ', Year of birth: '||info.year\_of\_birth || ', Place of birth: ' || info.place\_of\_birth );

END;



2. Write a PL/SQL block to print information about a given country.

a. Declare a PL/SQL record based on the structure of the countries table.

b. Define a variable countryid and pass the value througha substitution variable.

1. In the executable section, get all the information from the countries table by using countryid. Display selected information about the country. A sample output is shown below.
2. You may want to execute and test the PL/SQL block for the countries with the IDs DE, UK, US.

Изображение выглядит как текст, Шрифт, белый, снимок экрана

Автоматически созданное описание

DECLARE

type countries\_info is record(

country\_id varchar2(5),

country\_name varchar2(50),

region\_id number

);

my\_country\_id varchar2(5);

c\_info countries\_info;

BEGIN

my\_country\_id := 'CA';

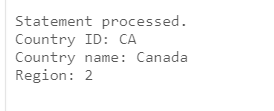
SELECT country\_id, country\_name, region\_id into c\_info from countries where country\_id = my\_country\_id;

DBMS\_OUTPUT.PUT\_LINE('Country ID: ' || c\_info.country\_id);

DBMS\_OUTPUT.PUT\_LINE('Country name: ' || c\_info.country\_name);

DBMS\_OUTPUT.PUT\_LINE('Region: ' || c\_info.region\_id);

END;



3. Within a PL/SQL block create an INDEX BY table to store the square of the numbers from 1 to 8. Output the result to the screen.

declare

type sq\_of\_numbers is table of number

index by pls\_integer;

squares pls\_integer;

sq\_table sq\_of\_numbers;

begin

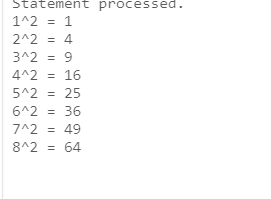
for squares in 1..8 loop

sq\_table(squares) := power(squares, 2);

DBMS\_OUTPUT.PUT\_LINE(squares || '^2 = ' || sq\_table(squares));

end loop;

end;



4. Create a PL/SQL block to retrieve the names of some departments from the departments table and print each department name on the screen, incorporating an INDEX BY table. Using a loop, retrieve the names of 10 departments and store the names in the INDEX BY table. Start with department\_id 10. Increase deptno by 10 for every iteration of the loop. The following table shows the department\_id for which you should retrieve the department\_name and store in the INDEX BY table. Using another loop, retrieve the department names from the INDEX BY table and display them.





declare

type departments\_table is table of departments.department\_name%type index by pls\_integer;

d\_index pls\_integer;

d\_table departments\_table;

begin

d\_index := 10;

while d\_index <=100 loop

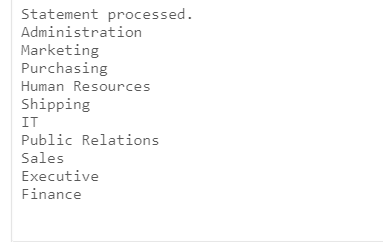
select department\_name into d\_table(d\_index) from departments where department\_id = d\_index;

DBMS\_OUTPUT.PUT\_LINE(d\_table(d\_index));

d\_index := d\_index + 10;

end loop;

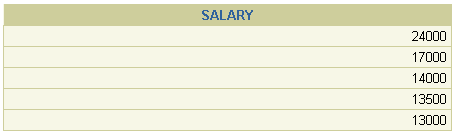
end;



**Practice 4.2 (Explicit Cursors)**

1. Create a PL/SQL block that determines the top *n* salaries of the employees.

* + - 1. Create a table top\_salaries with one column salary of the type NUMBER(8, 2).
      2. Declare a number *n* that represents the number of top *n* earners from the employees table.
      3. Declare a variable sal of type employees.salary.
      4. Declare a cursor, emp\_cursor, that retrieves the salaries of employees in descending order. Remember that the salaries should not be duplicated.
      5. In the executable sectio n, open the loop and fetch top *n* salaries and insert them into top\_salaries table. You can use a simple loop to operate on the data. Also, try and use %ROWCOUNT and %FOUND attributes for the exit condition.
      6. After inserting into the top\_salaries table, display the rows with a SELECT statement. The output shown represents the five highest salaries in the employees table.



* + - 1. Test a variety of special cases, such as *n* = 0 or where *n* is greater than the number of employees in the employees table. Empty the top\_salaries table after each test.

CREATE TABLE top\_salaries (

salary number

);

drop table top\_salaries;

declare

n number;

sal employees.salary%type;

cursor emp\_cursor is select salary from employees order by salary desc;

begin

n := 6;

open emp\_cursor;

for i in 1..n loop

fetch emp\_cursor into sal;

exit when emp\_cursor%rowcount >=n or emp\_cursor%NOTFOUND;

insert into top\_salaries(salary) values(sal);

end loop;

end;

select salary from top\_salaries;



2. Create a PL/SQL block that outputs all employees from the department 60 and with the salary more than 5000.

declare

cursor dep\_table is select last\_name from employees where department\_id = 60 and salary > 5000;

last\_name employees.last\_name%type;

begin

open dep\_table;

loop

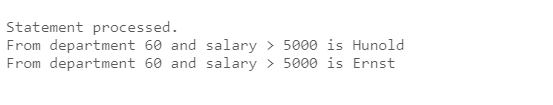
fetch dep\_table into last\_name;

exit when dep\_table%notfound;

DBMS\_OUTPUT.PUT\_LINE('From department 60 and salary > 5000 is ' || last\_name);

end loop;

end;



3. Rewrite the script of the task 2 using Cursor FOR Loop.

declare

cursor dep\_table is select last\_name from employees where department\_id = 60 and salary > 5000;

last\_name employees.last\_name%type;

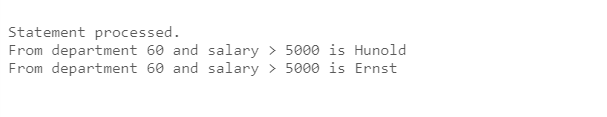
begin

for dep\_record in dep\_table loop

DBMS\_OUTPUT.PUT\_LINE('From department 60 and salary > 5000 is ' || dep\_record.last\_name);

end loop;

end;



4. Create a PL/SQL block that does the following:

* + - 1. Define a variable deptno to provide the department ID.
      2. Declare a cursor, emp\_cursor, that retrieves the last\_name, salary, and manager\_id of the employees working in the department specified in deptno.
      3. In the executable section use the cursor FOR loop to operate on the data retrieved. If the salary of the employee is less than 5000 and if the manager ID is either 101 or 124, display the message <<*last\_name*>> Due for a raise. Otherwise, display the message <<*last\_name*>> Not due for a raise.
      4. Test the PL/SQL block for the following cases:



declare

deptno departments.department\_id%type;

cursor emp\_cursor is select last\_name, salary, manager\_id from employees where department\_id = deptno;

begin

deptno:= 20;

for emp\_record in emp\_cursor loop

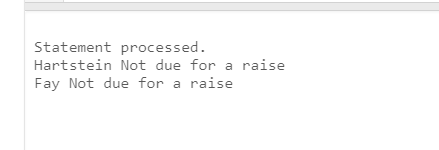
if (emp\_record.salary < 5000 and (emp\_record.manager\_id = 101 or emp\_record.manager\_id = 124)) then DBMS\_OUTPUT.PUT\_LINE(emp\_record.last\_name || ' Due for rise');

else DBMS\_OUTPUT.PUT\_LINE(emp\_record.last\_name || ' Not due for a raise');

end if;

end loop;

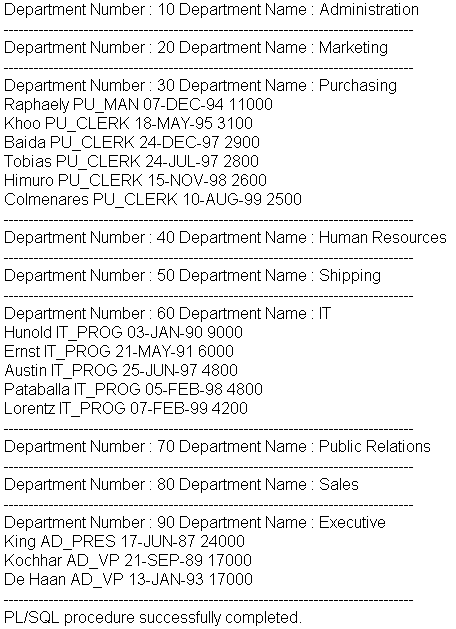
end;



5. Write a PL/SQL block, which declares and uses cursors with parameters. In a loop, use a cursor to retrieve the department number and the department name from the departments table for a department whose department\_id is less than 100. Pass the department number to another cursor as a parameter to retrieve from the employees table the details of employee last name, job, hire date, and salary of those employees whose employee\_id is less than 120 and who work in that department.

* + - 1. In the declarative section, declare a cursor dept\_cursor to retrieve department\_id, department\_name for those departments with department\_id less than 100. Order by department\_id.
      2. Declare another cursor emp\_cursor that takes the department number as parameter and retrieves last\_name, job\_id, hire\_date, and salary of those employees with employee\_id of less than 120 and who work in that department.
      3. Declare variables to hold the values retrieved from each cursor. Use the %TYPE attribute while declaring variables.
      4. Open the dept\_cursor, use a simple loop and fetch values into the variables declared. Display the department number and department name.
      5. For each department, open the emp\_cursor by passing the current department number as a parameter. Start another loop and fetch the values of emp\_cursor into variables and print all the details retrieved from the employees table. **Note:** You may want to print a lineafter you have displayed the details of each department.Use appropriate attributes for the exit condition. Also determine whether a cursor is already open before opening the cursor.
      6. Close all the loops and cursors, and end the executable section. Execute the script.

The sample output is shown below.



declare

cursor dept\_cursor is select department\_id, department\_name from departments where department\_id < 100;

cursor emp\_cursor (deptno number) is select last\_name, job\_id, hire\_date, salary from employees where employee\_id < 120 and department\_id = deptno;

dep\_id departments.department\_id%type;

dep\_name departments.department\_name%type;

emp\_last\_name employees.last\_name%type;

emp\_job\_id employees.job\_id%type;

emp\_hire\_date employees.hire\_date%type;

emp\_salary employees.salary%type;

begin

open dept\_cursor;

loop fetch dept\_cursor into dep\_id, dep\_name;

exit when dept\_cursor%notfound;

DBMS\_OUTPUT.PUT\_LINE('dep id: ' || dep\_id || ' dep name ' || dep\_name);

open emp\_cursor(dep\_id);

loop fetch emp\_cursor into emp\_last\_name, emp\_job\_id, emp\_hire\_date, emp\_salary;

exit when emp\_cursor%notfound;

DBMS\_OUTPUT.PUT\_LINE('Last name: ' || emp\_last\_name || 'Job id: ' || emp\_job\_id || 'Hire date: ' || emp\_hire\_date || 'Salary: ' || emp\_salary );

end loop;

close emp\_cursor;

end loop;

close dept\_cursor;

end;

