

PL/SQL

Control Structures

In addition to SQL commands, PL/SQL can also process data using flow of statements. The flow of control statements are classified into the following categories.

- Conditional control - Branching
- Iterative control - looping
- Sequential control

BRANCHING in PL/SQL:

Sequence of statements can be executed on satisfying certain condition.

If statements are being used and different forms of if are:

1. Simple IF
2. ELSIF
3. ELSE IF

SIMPLE IF:

Syntax:

IF condition THEN

statement1;

statement2;

END IF;

IF-THEN-ELSE STATEMENT:

Syntax:

IF condition THEN

statement1;

ELSE

statement2;

END IF;

ELSIF STATEMENTS:

Syntax:

IF condition1 THEN

statement1;

ELSIF condition2 THEN

statement2;

ELSIF condition3 THEN

statement3;

ELSE

statementn;

END IF;

NESTED IF :

Syntax:

IF condition THEN

statement1;

ELSE

IF condition THEN

statement2;

ELSE

statement3;

END IF;

END IF;

ELSE

statement3;

END IF;

SELECTION IN PL/SQL(Sequential Controls)

SIMPLE CASE

Syntax:

CASE SELECTOR

WHEN Expr1 THEN statement1;

WHEN Expr2 THEN statement2;

:

ELSE

Statement n;

END CASE;

SEARCHED CASE:

CASE

WHEN searchcondition1 THEN statement1;

WHEN searchcondition2 THEN statement2;

:

:

ELSE

statementn;

END CASE;

ITERATIONS IN PL/SQL

Sequence of statements can be executed any number of times using loop construct.

It is broadly classified into:

- Simple Loop
- For Loop
- While Loop

SIMPLE LOOP

Syntax:

LOOP

statement1;

EXIT [WHEN Condition];

END LOOP;

WHILE LOOP

Syntax:

WHILE condition LOOP

statement1;

statement2;

END LOOP;

FOR LOOP

Syntax:

FOR counter IN [REVERSE]

LowerBound..UpperBound

LOOP

statement1;

statement2;

END LOOP;

PROGRAM 1

Write a PL/SQL block to calculate the incentive of an employee whose ID is 110.

DECLARE

V_salary EMPLOYEES.SALARY % TYPE;
V_incentive NUMBER(10,2);

BEGIN

SELECT SALARY INTO V_salary

FROM EMPLOYEES ~~ID = 110;~~

WHERE EMPLOYEE_ID = 110;

V_incentive := V_salary * 0.10;

DBMS_OUTPUT.PUT_LINE('Incentive for Employee 110: ' || V_incentive);

EXCEPTION

WHEN NO_DATA_FOUND THEN

DBMS_OUTPUT.PUT_LINE('Employee ID 110 not found.');

WHEN OTHERS THEN

DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);

END;

PROGRAM 2

Write a PL/SQL block to show an invalid case-insensitive reference to a quoted and without quoted user-defined identifier.

BEGIN

EXECUTE IMMEDIATE 'CREATE TABLE test-table ("EmpName" VARCHAR2(50))

EXECUTE IMMEDIATE 'INSERT INTO test-table (empname) VALUES ('Alice')';

EXCEPTION

WHEN OTHERS THEN

DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);

END;

/



PROGRAM 3

Write a PL/SQL block to adjust the salary of the employee whose ID 122.

Sample table: employees

BEGIN

UPDATE employees

SET salary = salary * 1.05

(where employee-id = 122;

if SQL%ROWCOUNT > 0 then

commit;

DBMS_OUTPUT.PUT_LINE('salary adjusted');

end if;

END;

/

✓

PROGRAM 4

Write a PL/SQL block to create a procedure using the "IS [NOT] NULL Operator" and show AND operator returns TRUE if and only if both operands are TRUE.

DECLARE

V_name varchar(50) := 'John';

V_commission number := NULL;

BEGIN

if V_commission is NULL THEN

DBMS_output.put_line('Commission is NULL');

End if;

if (V_name is Not Null) AND (1=1) THEN

DBMS_output.put_line('AND operator returned True');

end if;

END;

/



PROGRAM 5

Write a PL/SQL block to describe the usage of LIKE operator including wildcard characters and escape character.

DECLARE

v_string varchar(30) := 'Test file - 20%';

BEGIN

-- % and - wild cards

if v_string LIKE 'Test %- %' THEN

DBMS_OUTPUT.put_line('Matches with % and -');

End if;

if v_string like '%20\%\' ESCAPE '\' THEN

DBMS_OUTPUT.put_line('Matches with % using escape');

end if;

END;

PROGRAM 6

Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num_small variable and large number will store in num_large variable.

DECLARE

num-a Number := 55;

num-b Number := 12;

num-small ~~Number~~;

num-large Number;

BEGIN

if num-a > num-b THEN

num-large := num-a;

num-small := num-b;

else

num-large := num-b;

num-small := num-a;

end if;

DBMS_OUTPUT.put_line('small: ' || num-small || 'large: ' ||
num-large);

END;

✓

PROGRAM 7

Write a PL/SQL procedure to calculate the incentive on a target achieved and display the message either the record updated or not.

BEGIN

if & target_achieved \geq 10000 THEN

UPDATE employee

SET incentive_pay = NVL(incentive_pay, 6) + 500

Where employee_id := & employee_id;

if SQL % Rowcount > 0 then

commit;

DBMS_OUTPUT.put_line('Record not update (ID not found);

End if;

else

DBMS_OUTPUT.put_line('Record No update, Target not);

End if;

End;

PROGRAM 8

Write a PL/SQL procedure to calculate incentive achieved according to the specific sale limit.

DECLARE

V_sale number := &man they - sales;

V_incentive number := 0;

BEGIN

if V_sales = 50000 then V_incentive := 2000;

elseif V_sales < 10000 then V_incentive := 500;

end if;

if V_incentive > 0 then

UPDATE employee SET incentive_pay = NVL(incentive_pay, 0) +

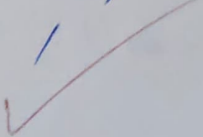
V_incentive where employee_id = &employee_id;

Commit;

end if;

DBMS_OUTPUT.put_line('Incentive Awarded: ' || V_incentive);

END;



PROGRAM 9

Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

DECLARE

v_current - emp NUMBER;

v_vacancies constant number := 45;

BEGIN

select count (*) into v_current - emp from employees
where department_id = 50;

if v_vacancies ~~available~~ > 0 then

DBMS_output.put_line('Dept 30 has' || v_current - emp
|| 'employees.');

DBMS_output.put_line('Result: yes,' || v_vacancies
|| 'vacancies available');

Else

DBMS_output.put_line('Result: No vacancies');

End if;

END;

PROGRAM 10

Write a PL/SQL program to count number of employees in a specific department and check whether this department have any vacancies or not. If any vacancies, how many vacancies are in that department.

DECLARE

```
v-dept-id number := & dept-id;  
v-capacity constant number := 60;  
v-current-emp number;  
v-vacancies number;
```

BEGIN

```
select count(*) into v-current-emp FROM employees  
where department-id = v-dept-id;
```

```
v-vacancies := v-capacity - v-current-emp;
```

```
DBMS_output.put_line('Employees in Dept ' || v-dept-id  
|| ' : ' || v-current-emp);
```

```
if v-vacancies > 0 then
```

```
DBMS_output.put_line('Result: yes', || v-vacancies ||  
'vacancies');
```

```
else
```

```
DBMS_output.put_line('Result: No vacancies or over capacity');
```

```
end if;
```

```
END;
```


PROGRAM 11

Write a PL/SQL program to display the employee IDs, names, job titles, hire dates, and salaries of all employees.

BEGIN

for r_emp IN

select e.employee_id, e.first_name || ' ' ||

e.last_name AS name, j.job_title, e.hire_date, e.salary

FROM employees e JOIN jobs j ON e.job_id = j.job_id

order by e.employee_id

)

LOOP

DBMS_Output.put_line(r_emp.employee_id || ' ' || r_emp.name

|| ' ' || r_emp.job_title || ' ' || r_emp.hire_date || ' ' ||

r_emp.salary);

END LOOP;

END;

/

PROGRAM 12

Write a PL/SQL program to display the employee IDs, names, and department names of all employees.

BEGIN

for r_emp IN

select e.employee_id, e.first_name || ' ' || e.last_name
as name, d.department_name

FROM employee e LEFT JOIN department ON

e.department_id = d.department_id ORDER BY

e.employee_id

)

LOOP

DBMS_OUTPUT.put_line(r_emp.employee_id

|| ' ' || r_emp.name || ' ' || r_emp.name || ' ' ||

r_emp.department_name);

END loop;

END;

/

PROGRAM 13

Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all jobs.

BEGIN

for r in job IN

SELECT job_id, job_title, min_salary

FROM jobs

ORDER by min_salary DESC

)

LOOP

DBMS_OUTPUT.put_line(r.job_id || ' ' ||

r.job_title || ' ' || r.min_salary);

END loop;

END;

/

✓

PROGRAM 14

Write a PL/SQL program to display the employee IDs, names, and job history start dates of all employees.

BEGIN

for r_hist IN (


select e.employee-id, e.first-name || ' ' || e.last-name
as name, jh.start-date from employee e join
job_history jh ON employee-id = jh.employee-id
ORDER by employee-id, jh.start-date
)

LOOP

DBMS_OUTPUT.put_line(r_hist.employee-id || ' ' ||
r_hist.name || ' | start | ' || r_hist.start-date);

END LOOP;

END;



PROGRAM 15

Write a PL/SQL program to display the employee IDs, names, and job history end dates of all employees.

BEGIN

for r_hist IN

select e.employee_id, e.first_name || ' ' ||
e.last_name AS name, jh.end_date from
employee e, employee_id = jh.employee_id
order by e.employee_id, jh.end_date

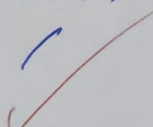
)

LOOP

DBMS_output.put_line(r_hist.employee_id || ' ' ||
r_hist.name || ' | End: ' || NVL(to_char(r_hist.end_date),
'N/A'));;

END LOOP;

END;



Evaluation Procedure	Marks awarded
PL/SQL Procedure(5)	5
Program/Execution (5)	5
Viva(5)	5
Total (15)	15
Faculty Signature	<i>Bml</i>