بسم الله الرحمن الرحيم

Database Programming

برمجة قواعد البيانات

أستاذات/المادة: م. لندا عمر البدري

م. نجلاء حسن

Control Statements

Conditional statements

- Conditional statements in PL/SQL have 3 variables:
  - if-then
  - if-then-else
  - if-then- elsif

Conditional statements

Conditional Statements	syntax

if-then	if <condition> then
	<statement-list>
	end if;

if-then-else	if <condition> then
	<statement-list-1>
	else
	<statement-list-2>
	end if;

if-then-elsif	if <condition-1> then
	<statement-list-1>
	elsif <condition-2>
	<statement-list-2>
	.....
	elsif <condition-N>
	<statement-list-N>
	else
	<statement-list-N+1>
	end if;

PL/SQL Decision Control Structures

Use IF/ELSIF to evaluate many conditions:

- IF condition1 THEN

commands that execute if condition1 is TRUE;

ELSIF condition2 THEN

commands that execute if condition2 is TRUE;

ELSIF condition3 THEN

commands that execute if condition3 is TRUE;

...

ELSE

commands that execute if none of the

conditions are TRUE;

END IF;

Conditional logic –IF statement

Examples

```
IF hourly_wage > 10  THEN
hourly_wage := hourly_wage * 1.5;
ELSE
hourly_wage := hourly_wage * 1.1;
END IF;

IF salary BETWEEN 1000AND 4000
THEN
bonus := 1500;
ELSIF salary > 4000 AND salary <= 10000
THEN bonus := 1000;
ELSE bonus := 0;
END IF;
```

Comments

- You can put parenthesis around boolean expression after the IF and ELSIF .
- You don't need to put {, } or BEGIN, END to surround several statements between IF and ELSIF/ELSE, or between ELSIF/ELSE and END IF;

Example

1)

```
if (cnum > 1000) and (cnum < 9000) then
dbms_output.put_line('Customer no ' || cnum);
end if;
```

2)

```
if (cnum > 1000) and (cnum < 9000) then
i := i+1;
dbms_output.put_line(' Valid Customer ' || cnum);
else
j := j+1;
dbms_output.put_line('Invalid Customer ' || cnum);
end if;
```

Example (cont.)

3)

```
if (score > 90) then
na := na+1;
elsif (score > 80) then
nb := nb+1;
elsif (score > 70) then
nc := nc+1;
elsif (score > 60) then
nd := nd+1;
else
nf := nf+1;
end if;
```

IF/ELSIF Example

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> DECLARE
  2     todays_date DATE;
  3     current_day VARCHAR2(9);
  4  BEGIN
  5     todays_date := SYSDATE;
  6     -- extract day portion from current date, and trim trailing blank spaces
  7     current_day := TO_CHAR(todays_date, 'DAY');
  8     current_day := INITCAP(current_day);
  9     current_day := RTRIM(current_day);
 10     -- IF/ELSIF condition to determine current day
 11     IF current_day = 'Friday' THEN
 12       DBMS_OUTPUT.PUT_LINE('Today is Friday!');
 13     ELSIF current_day = 'Saturday' THEN
 14       DBMS_OUTPUT.PUT_LINE('Today is Saturday!');
 15     ELSIF current_day = 'Sunday' THEN
 16       DBMS_OUTPUT.PUT_LINE('Today is Sunday!');
 17     ELSIF current_day = 'Monday' THEN
 18       DBMS_OUTPUT.PUT_LINE('Today is Monday!');
 19     ELSIF current_day = 'Tuesday' THEN
 20       DBMS_OUTPUT.PUT_LINE('Today is Tuesday!');
 21     ELSIF current_day = 'Wednesday' THEN
 22       DBMS_OUTPUT.PUT_LINE('Today is Wednesday!');
 23     ELSIF current_day = 'Thursday' THEN
 24       DBMS_OUTPUT.PUT_LINE('Today is Thursday!');
 25     ELSE
 26       DBMS_OUTPUT.PUT_LINE('Current day not found.');
 27     END IF;
 28  END;
 29  /
Today is Tuesday!

PL/SQL procedure successfully completed.
```

Add/modify these commands

Complex Conditions

- Created with logical operators AND, OR and NOT
- AND is evaluated before OR
- Use () to set precedence

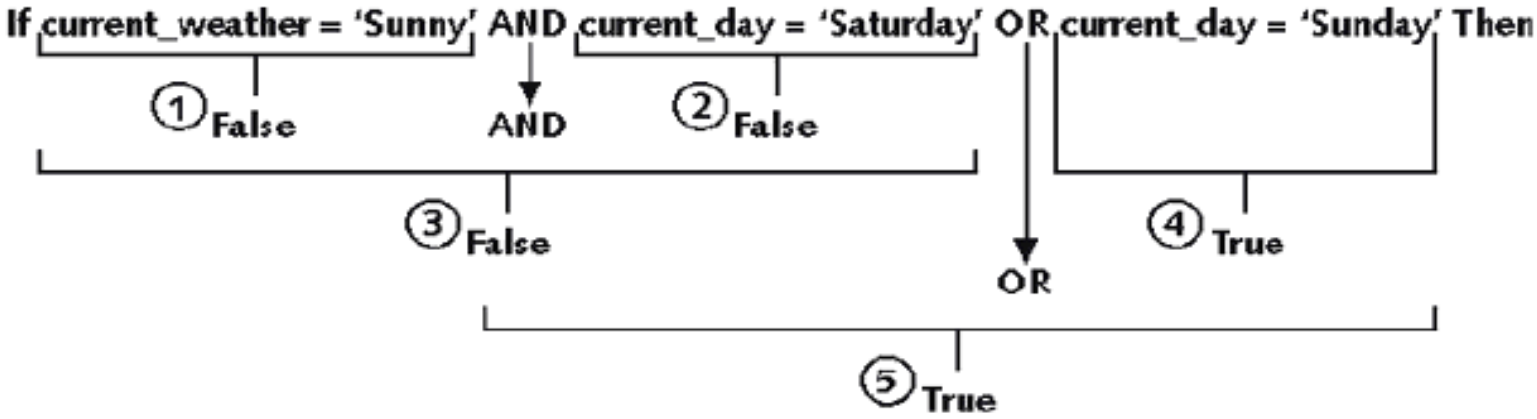

Figure 4-19 Evaluating AND and OR in an expression

Conditional logic

Condition:

```
If <cond>
   then <command>
elsif <cond2>
   then <command2>
else
      <command3>
end if;
```

Nested conditions:

```
If <cond>
   then
      if <cond2>
         then
               <command1>
       end if;
 else <command2>
 end if;
```

IF-THEN-ELSIF Statements

```
. . .
IF rating > 7 THEN
   v_message := 'You are great';
ELSIF rating >= 5 THEN
   v_message := 'Not bad';
ELSE
   v_message := 'Pretty bad';
END IF;
. . .
```

Suppose we have the following table:

```
create table mylog(
who varchar2(30),
logon_num number
);
```

mylog

who	logon_num
Hala	3
Amal	4
Mona	2

- Want to keep track of how many times someone logged on to the DB
- When running, if user is already in table, increment logon_num. Otherwise, insert user into table

Solution

```
DECLARE
   cnt  NUMBER;
BEGIN
   select count(*)
    into cnt
    from mylog
   where who = user;

   if cnt > 0 then
     update mylog
       set logon_num = logon_num + 1
     where who = user;
   else
     insert into mylog values(user, 1);
   end if;
   commit;
end;
/
```

Conditional logic –Simple CASE statement

CASE *selector*
WHEN *expression_*1 THEN *statements*
[WHEN *expression_*2 THEN *statements*]
[ELSE *statements*]
END CASE;

- *selector* can be an expression of any data type, and it provides the value we are comparing.
- *Expression_n* is the expression to test for equality with the selector.

- If no WHEN matches the selector value, then the ELSE clause is executed.
- If there is no ELSE clause PL/SQL will implicitly supply:

ELSE RAISE CASE_NOT_FOUND;

which will terminate the program with an error (if the program ends up in the ELSE clause).

```
CASE grade
   WHEN 'A' THEN
    dbms_output.put_line('Excellent');
   WHEN 'B' THEN
    dbms_output.put_line('Very Good');
   WHEN 'C' THEN
    dbms_output.put_line('Good');
   WHEN 'D' THEN    dbms_output.put_line('Fair');
   WHEN 'F' THEN
         dbms_output.put_line('Poor');
   ELSE dbms_output.put_line('No such grade');
END CASE;
```

Loops

- Program structure that executes a series of program statements, and periodically evaluates an exit condition to determine if the loop should repeat or exit
- Pretest loop: evaluates the exit condition before any program commands execute
- Posttest loop: executes one or more program commands before the loop evaluates the exit condition for the first time
- PL/SQL has 5 loop structures

The LOOP…EXIT Loop

```
LOOP
  [program statements]
  IF condition THEN
    EXIT;
  END IF;
  [additional program statements]
END LOOP
```

The LOOP…EXIT Loop

```
create table number_table(
      num NUMBER(10)
);
```

```
DECLARE
i       number_table.num%TYPE:=1;
BEGIN
   loop
   IF i > 10 THEN EXIT;
       END IF;
      INSERT INTO number_table
      VALUES(i);
      i := i + 1;
   END LOOP;
END;
```

The LOOP…EXIT WHEN Loop

```
LOOP
    program statements
    EXIT WHEN condition;
END LOOP;
```

The LOOP…EXIT WHEN Loop

```
DECLARE
i       number_table.num%TYPE:=1;
BEGIN
loop
        INSERT INTO number_table
      VALUES(i);
      i := i + 1;
EXIT WHEN i > 10 ;
END LOOP;
END;
```

The WHILE...LOOP

WHILE *condition* LOOP

program statements

END LOOP;

Loops: WHILE Loop

```
DECLARE
i       number_table.num%TYPE:=1;
BEGIN
   WHILE i <= 10 LOOP
       INSERT INTO number_table
       VALUES(i);
       i := i + 1;
   END LOOP;
END;
```

The Numeric FOR Loop

FOR *counter_variable* IN
  start_value .. end_value
LOOP
    program statements
END LOOP;

Loops: FOR Loop

```
DECLARE
   i        number_table.num%TYPE;
BEGIN
   FOR i IN 1..10 LOOP
      INSERT INTO number_table VALUES(i);
   END LOOP;
END;
```

Notice that i is incremented automatically

- Lastly the Cursor FOR Loop , which we will discuss later

THE END