Conditionals and Loops

Case statement

- Simple case statement
- Searched case statement

Simple case statement

```
case {expr}
    when {match1} then
        statement1;
    [when {match2} then
        statement2;]
    [....]
    [else
        statementN;]
end case;
```

Search case statement

```
case
    when {condition1} then
        statement1;
    [when {condition2} then
        statement2;]
    [...]
    [else
        statementN;]
end case
```

GOTO statements

- During the execution of a PL/SQL code, you can jump to a different part of the code, using the GOTO statement.

  ```
  goto label
  ```

- Here the `label` is a location in your code, marked with `<<label>>`
- In modern language like Java or C#, use of `goto` is disabled.
- Even in C language which introduced this concept, it is highly discouraged.
- While you can goto any part of the code, there are few restrictions
  - GOTO cannot transfer control to a statement inside a `if` or `case` or `loop` blocks.

Loops

- just a `loop`
- for loop
- while loop
- goto as a looping statement

```
<<the_loop>>
loop
    --statements
end loop the_loop;

-- or just

loop
    --statements
end loop;
```

You can use the `exit` command to come out of a running loop, usually done conditionally.

```
loop
    -- statements
    if condition1 then
        exit;
    end if;
end loop;

--or just

loop
    -- statements
    exit when condition;
end loop;
```

For example,

```
declare
    i number := 1;
begin
    loop
        dbms_output.put_line('i is ' || i);
        i := i+1;
        if i>10 then
            goto the_end;
        end if;
    end loop;

    <<the_end>>
    null;
end;
```

```
----------------------------------------
declare
    i number := 1;
begin
    loop
        dbms_output.put_line('i is ' || i);
        i := i+1;
        if i>10 then
            exit;
        end if;
    end loop;
end;
----------------------------------------
declare
    i number := 1;
begin
    loop
        dbms_output.put_line('i is ' || i);
        i := i+1;
        exit when i>10;
    end loop;
end;
```

WHILE loop

Used:

- When the loop execution is based on a condition
- When we don't know the number of iterations to be performed

Syntax:

```
WHILE {condition} LOOP
    --STATEMENTS;
END LOOP;
```

FOR loop

Used when:

- we know the number of iterations in advance

Syntax:

```
FOR {loop_var} IN [REVERSE] {lower_limit}..{upper_limit} LOOP
    -- statements;
END LOOP;
```

- the loop variable is automatically declared and initialized to the `lower_limit`
- Even if you have a variable with the same name as the loop variable, it will be used as a loop variable

Both the `while` and `for` loops can be used with `CURSORS`.