# IF ELSE, CASE, LOOP, CURSOR in PLSQL

G. M. Shahariar
Lecturer
Department of CSE
Ahsanullah University of Science & Technology

# Problem

Check if a number is even.

If the number is even, print EVEN.

Use IF...THEN syntax to solve the problem.

# “IF…THEN” Syntax

```
SET SERVEROUTPUT ON

DECLARE
        ……..
BEGIN
      IF …….. THEN

            ……..

      END IF;
END;
/
```

# “IF…THEN” Syntax

```
SET SERVEROUTPUT ON


DECLARE
        NUM number := 20;
BEGIN
    IF MOD(NUM, 2) = 0 THEN
        DBMS_OUTPUT.PUT_LINE(‘EVEN’);
    END IF;
END;
/
```

# Problem

Check whether a number is even or odd.

If the number is Even, print EVEN.
If the number is Odd, print ODD.

Use IF...THEN...ELSE syntax to solve the problem.

# “IF…THEN ELSE” Syntax

```
SET SERVEROUTPUT ON

DECLARE
        ……..
BEGIN
      IF ……..  THEN

            ……..
      ELSE
            ……..

      END IF;
END;
/
```

# “IF…THEN ELSE” Syntax

```
SET SERVEROUTPUT ON

DECLARE
        NUM number := 21;
BEGIN
      IF MOD(NUM, 2) = 0 THEN

            DBMS_OUTPUT.PUT_LINE(‘EVEN’);
      ELSE
            DBMS_OUTPUT.PUT_LINE(‘ODD’);
      END IF;
END;
/
```

# Problem

Mod a number by 3.

There can be three possible results –

If the result is 0, print ZERO
If the result is 1, print ONE
If the result is 2, print TWO

Use IF..THEN..ELSIF...THEN...ELSE syntax to solve the problem.

# “IF…THEN…ELSIF….THEN…ELSE” Syntax

```
SET SERVEROUTPUT ON
DECLARE
        ……..
BEGIN
        IF …….. THEN
                ……..
        ELSIF…….. THEN
                ……..
        ELSE
                ……..
        END IF;
END;
/
```

# “IF…THEN…ELSIF….THEN…ELSE” Syntax

```
SET SERVEROUTPUT ON
DECLARE
    NUM number := 21;
BEGIN
    IF MOD(NUM, 3) = 0 THEN
        DBMS_OUTPUT.PUT_LINE(‘ZERO’);
    ELSIF MOD(NUM, 3) = 1 THEN
        DBMS_OUTPUT.PUT_LINE(‘ONE’);
    ELSE
        DBMS_OUTPUT.PUT_LINE(‘TWO’);
    END IF;
END;
/
```

# Problem

Mod a number by 3.

There can be three possible results –

If the result is 0, print ZERO
If the result is 1, print ONE
If the result is 2, print TWO

Use CASE…WHEN…THEN…ELSE syntax to solve the problem.

# “CASE…WHEN…THEN…ELSE” Syntax

```
SET SERVEROUTPUT ON
DECLARE
        ……..
BEGIN
        CASE
            WHEN …….. THEN
                    ……..
            WHEN …….. THEN
                    ……..
            ELSE
                    ……..
        END CASE;
END;
/
```

# “CASE…WHEN…THEN…ELSE” Syntax

```
SET SERVEROUTPUT ON
DECLARE
    NUM number := 20;
BEGIN
    CASE
        WHEN MOD(NUM, 3) = 0 THEN
            DBMS_OUTPUT.PUT_LINE(‘ZERO’);
        WHEN MOD(NUM, 3) = 0 THEN
            DBMS_OUTPUT.PUT_LINE(‘ONE’);
        ELSE
            DBMS_OUTPUT.PUT_LINE(‘TWO’);
    END CASE;
END;
/
```

π

# Another CASE…WHEN…THEN…ELSE Syntax !

# “CASE…WHEN…THEN…ELSE” Syntax

```
SET SERVEROUTPUT ON
DECLARE
        ……..
BEGIN
        CASE ……..
            WHEN …….. THEN
                    ……..
            WHEN …….. THEN
                    ……..
            ELSE
                    ……..
        END CASE;
END;
/
```

# “CASE…WHEN…THEN…ELSE” Syntax

```
SET SERVEROUTPUT ON
DECLARE
    NUM number := 19;
BEGIN
    CASE MOD(NUM, 3)
        WHEN 0 THEN
            DBMS_OUTPUT.PUT_LINE(‘ZERO’);
        WHEN 1 THEN
            DBMS_OUTPUT.PUT_LINE(‘ONE’);
        ELSE
            DBMS_OUTPUT.PUT_LINE(‘TWO’);
    END CASE;
END;
/
```

# Problem

Print 1 2 3 4 5.

Use LOOP, WHILE LOOP, FOR LOOP.

Observe the breaking condition in each case.

# “PL/SQL LOOP” Syntax

```
SET SERVEROUTPUT ON
DECLARE
        ……..
BEGIN
    LOOP
        ……
        ……
        IF ……THEN
            EXIT;
        END IF;

    END LOOP;
END;
/
```

# “PL/SQL LOOP” Syntax

```
SET SERVEROUTPUT ON
DECLARE
        NUM number := 0;
BEGIN
    LOOP
        NUM := NUM + 1;
        DBMS_OUTPUT.PUT_LINE(NUM);
        IF NUM = 5 THEN
            EXIT;
        END IF;

    END LOOP;
END;
/
```

# “PL/SQL LOOP” Syntax

```
SET SERVEROUTPUT ON
DECLARE
        ……..
BEGIN
    LOOP
        ……..
        ……..
        EXIT WHEN ……..
    END LOOP;
END;
/
```

# “PL/SQL LOOP” Syntax

```
SET SERVEROUTPUT ON

DECLARE
        NUM number := 0;
BEGIN
    LOOP
        NUM := NUM + 1;
        DBMS_OUTPUT.PUT_LINE(NUM);
        EXIT WHEN NUM = 5;
    END LOOP;
END;
/
```

# “WHILE LOOP” Syntax

```
SET SERVEROUTPUT ON
DECLARE
        ……..
BEGIN
    WHILE ……..
    LOOP
        ……..
        ……..
    END LOOP;
END;
/
```

# “WHILE LOOP” Syntax

```
SET SERVEROUTPUT ON
DECLARE
        NUM number := 0;
BEGIN
    WHILE NUM < 5
    LOOP
        NUM := NUM + 1;
        DBMS_OUTPUT.PUT_LINE(NUM);
    END LOOP;
END;
/
```

# “FOR LOOP” Syntax

```
SET SERVEROUTPUT ON

DECLARE

            ……..

BEGIN

      FOR…….  IN ……..  LOOP

            ……..

            ……..

      END LOOP;

END;

/
```

# “FOR LOOP” Syntax

```
SET SERVEROUTPUT ON

DECLARE
        NUM number := 5;
BEGIN
    FOR i IN 1..NUM LOOP
        DBMS_OUTPUT.PUT_LINE(i);
    END LOOP;
END;
/
```

# PL/SQL CURSOR

A cursor is a pointer that points to a result of a query. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the **active set**.

You can name a cursor so that it could be referred to in a program to fetch and process the rows returned by the SQL statement, one at a time.

# CURSOR Types

Two Types –

1. Implicit Cursor: automatically created by Oracle whenever an SQL statement is executed. Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

2. Explicit Cursor: An explicit cursor should be defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row.

Working with an explicit cursor includes the following steps –

- Declaring the cursor for initializing the memory
- Opening the cursor for allocating the memory
- Fetching the cursor for retrieving the data
- Closing the cursor to release the allocated memory

FIRST YOU HAVE TO RUN THE PROVIDED ‘1.sql’ FILE.

```
SET SERVEROUTPUT ON
DECLARE
        ……
        ……
        CURSOR …… IS
        ……
BEGIN
      OPEN ……
          LOOP
              FETCH …… INTO ……
              EXIT WHEN ……
              ……
          END LOOP;

      CLOSE ……
END;
/
```

```
SET SERVEROUTPUT ON
DECLARE
        A money.id%TYPE;
        B money.taka%TYPE;
        CURSOR Hello IS
            SELECT id, taka from money;
BEGIN
       OPEN Hello;
           LOOP
               FETCH Hello INTO A, B;
               EXIT WHEN Hello%notfound;
               DBMS_OUTPUT.PUT_LINE(A || ' ' || B);
           END LOOP;
       CLOSE Hello;
END;
/
```

```
SET SERVEROUTPUT ON
DECLARE
        ……
        ……
        CURSOR …… IS
        ……
BEGIN
      OPEN ……
          FOR …… IN …… LOOP
                FETCH …… INTO ……
                ……
          END LOOP;

      CLOSE ……
END;
/
```

```
SET SERVEROUTPUT ON
DECLARE
        A money.id%TYPE;
        B money.taka%TYPE;
        CURSOR Hello IS
            SELECT id, taka from money;
BEGIN
      OPEN Hello;
          FOR i IN 1..2 LOOP
              FETCH Hello INTO A, B;
              DBMS_OUTPUT.PUT_LINE(A || ' ' || B);
          END LOOP;

      CLOSE Hello;
END;
/
```

```
SET SERVEROUTPUT ON
DECLARE
        ……..
        ……..
        CURSOR ……  (……..) IS

            ……..
BEGIN
      OPEN ……..
          LOOP
              FETCH …….  INTO ……..
              EXIT WHEN ……..
              ……..
          END LOOP;

      CLOSE ……
END;
/
```

```
SET SERVEROUTPUT ON
DECLARE
        A money.id%TYPE;
        B money.taka%TYPE;
        CURSOR Hello (nid money.id%TYPE) IS
            SELECT id, taka from money WHERE id > nid;
BEGIN
      OPEN Hello(1);
          LOOP
              FETCH Hello INTO A, B;
              EXIT WHEN Hello%notfound;
              DBMS_OUTPUT.PUT_LINE(A || ' ' || B);
          END LOOP;
      CLOSE Hello;
END;
/
```