File Edit View History Bookmarks Tools Help

Introduction to PL/SQL

www.pgrocer.net/Cis50/introplsql.html

Introduction to PL/SQL

PL/SQL is a procedural language that can use a step through records approach to handle processing. It implements IF statements and loops, variables and types, procedures and functions while still allowing for traditional query and maintenance type processing through SQL. The procedural constructions available in PL/SQL add tremendous power to the processing capabilities available with the Oracle relational database.

PL/SQL is written in a block structure where each block performs a logical task. Blocks can be nested within each other. The structure of a block is:

DECLARE (optional)
    Declarative section - allows for variables, types, cursors, user defined-exceptions etc
BEGIN (required)
    Executable section - PL/SQL procedural statements and SQL statements
EXCEPTION (optional)
    Exception section - error and exception handling
END; (required)

Example:

In this example, I declared a message called v_msg in the declarative section. Then I assigned a literal to v_msg. The assign is done using the assignment code :=. This means that the data to the right of the assignment sign will be assigned to the dataname on the left of the assignment sign.This ended the PL/SQL block.

Looking at the code below, the edit afst_intro takes me into the editor where I coded the PL/SQL code. When I had completed the code and saved it, I exited the editor and ran the code. This can be done with START followed by the name of the code or @ followed by the name of the code. In this case it would be START afst_intro or @ afst_intro.

SQL CODE:

```
SQL> EDIT afst_intro
```

PL/SQL CODE:

```
DECLARE
  v_msg VARCHAR2(25);
BEGIN
  v_msg :='This PL/SQL block works!';
END;
/
```

SQL CODE:

```
SQL> @ afst_intro

PL/SQL procedure successfully completed.
```

If I want to be able to see the message and assure myself that it works, I can use the following code to show the output. If you set serveroutput on in SQL*PLUS (notice you also set it off) then you can display output on the screen using dbms_output.put_line and enclosing the information you want to see within quotes. In the example below, I have a literal message followed by the contents of a PL/SQL defined variable.

SQL CODE:

Type here to search

9:39 AM
10/22/2019

Title: Oct 22-9:31 AM (1 of 29)

File Edit View History Bookmarks Tools Help

Introduction to PL/SQL

www.pgrocer.net/Cis50/introplsql.html

```
SQL> @ afst_intro

PL/SQL procedure successfully completed.
```

If I want to be able to see the message and assure myself that it works, I can use the following code to show the output. If you set serveroutput on in SQL*PLUS (notice you also set it off) then you can display output on the screen using dbms_output.put_line and enclosing the information you want to see within quotes. In the example below, I have a literal message followed by the contents of a PL/SQL defined variable.

SQL CODE:

```
SQL> edit afst1_intro
```

PL/SQL CODE:

```
SET SERVEROUTPUT ON
DECLARE
  v_msg1 VARCHAR2(25);
BEGIN
  v_msg1 :='This PL/SQL block works!';
  DBMS_OUTPUT.PUT_LINE('The output is: '|| v_msg1);
END;
/
SET SERVEROUTPUT OFF
```

SQL CODE:

```
SQL> edit afst1_intro
SQL> @ afst1_intro
The output is: This PL/SQL block works!

PL/SQL procedure successfully completed.
```

This example could also have been done by declaring the variable within SQL*Plus instead of within PL/SQL. When you declare the variable within SQL*Plus, you can use the PRINT followed by the field you want to see to display information on the screen.

I wanted to see the results and make sure the message was in v_msg so I added the PRINT line telling it to display the contents of v_msg so you could see how it worked. This command works after a declaration in SQL*Plus and assignment in PL/SQL. It cannot be done if v_msg is declared within the PL/SQL block. This line can be executed at the SQL> prompt or it can be added after the / in the code in the editor. Throughout this session, I can access the variable and give the command PRINT v_msg and see the results.

The :v_msg inside the begin is necessary because v_msg was declared externally rather then within the PL/SQL block. To reference a bind variable which is a variable that is declared outside the PL/SQL block, you need to have a colon in front of the name when you refer to it within the PL/SQL block.

SQL CODE:

```
SQL> VARIABLE v_msg VARCHAR2(25);
SQL> edit bfst_intro
```

PL/SQL CODE:

```
BEGIN
  :v_msg :='This PL/SQL still works!';
END;
```

Type here to search

9:41 AM 10/22/2019

File Edit View History Bookmarks Tools Help

Introduction to PL/SQL

www.pgrocer.net/Cis50/introplsql.html

SQL CODE:

```
SQL> VARIABLE v_msg VARCHAR2(25);
SQL> edit bfst_intro
```

PL/SQL CODE:

```
BEGIN
  :v_msg :='This PL/SQL still works!';
END;
/
SQL CODE:

SQL> START bfst_intro;

PL/SQL procedure successfully completed.

SQL> print v_msg

V_MSG
--------------------------------
This PL/SQL still works!


Or the variable assignment and/or the print can be included in the editor code, although it
is still not inside the block.  Notice the :v_msg inside the block, this is necessary
because v_msg is declared externally.
```

SQL CODE:

```
SQL> EDIT fst_intro
```

PL/SQL CODE:

```
VARIABLE v_msg VARCHAR2(25)
BEGIN
  :v_msg :='This PL/SQL works again!';
END;
/
PRINT v_msg
```

SQL CODE:

```
SQL> START fst_intro

PL/SQL procedure successfully completed.
```

Type here to search

9:44 AM
10/22/2019

File Edit View History Bookmarks Tools Help

Introduction to PL/SQL

www.pgrocer.net/Cis50/introplsql.html

```
SQL> edit snd_intro
```

PL/SQL CODE:

```
SET SERVEROUTPUT ON
DECLARE
  v_idno donornew.idno%TYPE;
  v_name donornew.name%TYPE;
  v_yrgoal donornew.yrgoal%TYPE;
BEGIN
  SELECT idno, name, yrgoal
  INTO v_idno, v_name, v_yrgoal
  FROM donornew
  WHERE idno = '12121';
  dbms_output.put_line (v_idno||'  '||v_name||'  '||v_yrgoal);
END;
/
SET SERVEROUTPUT OFF
```

This is a great way of getting the current type - you tie it to the description on the table.

SQL CODE:

```
SQL> @ snd_intro
12121  Jennifer Ames  440

PL/SQL procedure successfully completed.
```

In the example below, instead of having the identification number a constant in the code, I want the user to enter the identificaiton number and then I want to retrieve the matching record. I will illustrate two ways to do this below. The first way takes in the input within the BEGIN and the second takes it in from SQL*Plus ACCEPT statement.

SQL CODE:

```
SQL> edit snd1_intro
```

PL/SQL CODE:

```
SET SERVEROUTPUT ON
DECLARE
  v_idno donornew.idno%TYPE;
  v_name donornew.name%TYPE;
  v_yrgoal donornew.yrgoal%TYPE;
```

Type here to search

9:47 AM
10/22/2019

File Edit View History Bookmarks Tools Help

Introduction to PL/SQL

www.pgrocer.net/Cis50/introplsql.html

```
SQL> edit snd1_intro
```

PL/SQL CODE:

```
SET SERVEROUTPUT ON
DECLARE
  v_idno donornew.idno%TYPE;
  v_name donornew.name%TYPE;
  v_yrgoal donornew.yrgoal%TYPE;
  v_in_idno donornew.idno%TYPE;
BEGIN
  v_in_idno := &input_idno;
  SELECT idno, name, yrgoal
  INTO v_idno, v_name, v_yrgoal
  FROM donornew
  WHERE idno = v_in_idno;
  dbms_output.put_line (v_idno||'  '||v_name||'  '||v_yrgoal);
END;
/
SET SERVEROUTPUT OFF
```

from file/table

hold user input

SQL CODE:

```
SQL> START snd1_intro
Enter value for input_idno: 11111
old   7:   v_in_idno := &input_idno;
new   7:   v_in_idno := 11111;
11111  Stephen Daniels  650

PL/SQL procedure successfully completed.
```

SQL CODE:

```
SQL> edit snd2_intro
```

PL/SQL CODE:

```
SET SERVEROUTPUT ON
ACCEPT input_idno PROMPT 'Enter the identification number: '
DECLARE
  v_idno donornew.idno%TYPE;
  v_name donornew.name%TYPE;
```

Type here to search

9:54 AM
10/22/2019

```
SQL> edit snd2_intro
```

PL/SQL CODE:

```
SET SERVEROUTPUT ON
ACCEPT input_idno PROMPT 'Enter the identification number: '
DECLARE
  v_idno donornew.idno%TYPE;
  v_name donornew.name%TYPE;
  v_yrgoal donornew.yrgoal%TYPE;
 BEGIN
  SELECT idno, name, yrgoal
  INTO v_idno, v_name, v_yrgoal
  FROM donornew
  WHERE idno = &input_idno;
  dbms_output.put_line (v_idno||'  '||v_name||'  '||v_yrgoal);
END;
/
SET SERVEROUTPUT OFF
```

SQL CODE:

```
SQL> @ snd2_intro
Enter the identification number: 11111
old   9:   WHERE idno = &input_idno;
new   9:   WHERE idno = 11111;
11111  Stephen Daniels  650

PL/SQL procedure successfully completed.
```

Nesting blocks:

We will come back to this section.

The next topic will deal with a brief introduction to nesting blocks within each other and the scope of the variables defined in the blocks.

SQL CODE:

```
SQL> edit nest1_intro
```

I defined v_first and v_msg in the outer block and in the inner block. I gave them initial values in both places. In the outer block v_first has an initial value of 212 and v_msg has an initial value of * Outer define *. In the inner block v_msg has an initial value of 555 and v_msg has an initial value of * Inner define *. The things defined in the outer

Title: Oct 22-9:57 AM (6 of 29)

File Edit View History Bookmarks Tools Help

PL/SQL IF statements

www.pgrocer.net/Cis50/plsqlif1.html

SQL> edit donor0

PL/SQL CODE:

Starting IF statements.

```
SET VERIFY OFF
DECLARE
   v_idno    VARCHAR2(5) := &input_idno;
   v_yrgoal  NUMBER(7,2);
BEGIN
   SELECT yrgoal INTO v_yrgoal
   FROM donornew
   WHERE idno = v_idno;
   IF v_yrgoal > 250 THEN
      v_yrgoal := v_yrgoal * 1.1;
   END IF;
   UPDATE donornew
   SET yrgoal = v_yrgoal
   WHERE idno = v_idno;
   COMMIT;
   END;
/
SET VERIFY ON
```

To execute the code, I can either enter START donor0 or @ donor0. When prompted for the idno, I entered 12121. The processing was done and the successful message was returned. To verify the processing I then did a select to show the file and indeed record 12121 was increased from 400 to 440.

SQL CODE:

```
SQL> @ donor0
Enter value for input_idno: 12121

PL/SQL procedure successfully completed.

Input truncated to 13 characters

SQL> SELECT * FROM donornew;

IDNO  NAME            STADR           CITY       ST ZIP   DATEFST      YRGOAL CONTACT
----- --------------- --------------- ---------- -- ----- --------- --------- ------------
11111 Stephen Daniels 123 Elm St      Seekonk    MA 02345 03-JUL-98       500 John Smith
12121 Jennifer Ames   24 Benefit St   Providence RI 02045 24-MAY-97       440 Susan Jones
22222 Carl Hersey     24 Benefit St   Providence RI 02045 03-JAN-98           Susan Jones
23456 Susan Ash       21 Main St      Fall River MA 02720 04-MAR-92       120 Amy Costa
33333 Nancy Taylor    26 Oak St       Fall River MA 02720 04-MAR-92        50 John Adams
34567 Robert Brooks   36 Pine St      Fall River MA 02720 04-APR-98        50 Amy Costa

6 rows selected.
```

I then reran the PL/SQL code and input an id where the goal was not > 250. As you can see, no change was made to the table.

SQL CODE:

```
SQL> @ donor0
Enter value for input_idno: 34567
```

Type here to search

9:58 AM
10/22/2019

File Edit View History Bookmarks Tools Help

PL/SQL IF statements

www.pgrocer.net/Cis50/plsqlif1.html

SQL CODE:

```
SQL> edit donor0
```

PL/SQL CODE:

```
SET VERIFY OFF
DECLARE
   v_idno     VARCHAR2(5) := &input_idno;
   v_yrgoal   NUMBER(7,2);
BEGIN
   SELECT yrgoal INTO v_yrgoal
   FROM donornew
   WHERE idno = v_idno;
   IF v_yrgoal > 250 THEN
      v_yrgoal := v_yrgoal * 1.1;
   END IF;
   UPDATE donornew
   SET yrgoal = v_yrgoal
   WHERE idno = v_idno;
   COMMIT;
   END;
/
SET VERIFY ON
```

To execute the code, I can either enter START donor0 or @ donor0. When prompted for the idno, I entered 12121. The processing was done and the successful message was returned. To verify the processing I then did a select to show the file and indeed record 12121 was increased from 400 to 440.

SQL CODE:

~~400~~
440

```
SQL> @ donor0
Enter value for input_idno: 12121

PL/SQL procedure successfully completed.

Input truncated to 13 characters

SQL> SELECT * FROM donornew;

IDNO  NAME            STADR           CITY       ST ZIP   DATEFST      YRGOAL CONTACT
----- --------------- --------------- ---------- -- ----- --------- --------- ------------
11111 Stephen Daniels 123 Elm St      Seekonk    MA 02345 03-JUL-98       500 John Smith
12121 Jennifer Ames   24 Benefit St   Providence RI 02045 24-MAY-97       440 Susan Jones
22222 Carl Hersey     24 Benefit St   Providence RI 02045 03-JAN-98           Susan Jones
23456 Susan Ash       21 Main St      Fall River MA 02720 04-MAR-92       120 Amy Costa
33333 Nancy Taylor    26 Oak St       Fall River MA 02720 04-MAR-92        50 John Adams
34567 Robert Brooks   36 Pine St      Fall River MA 02720 04-APR-98        50 Amy Costa

6 rows selected.
```

I then reran the PL/SQL code and input an id where the goal was not > 250. As you can see, no change was made to the table.

SQL CODE:

```
SQL> @ donor0
```

Type here to search

10:03 AM 10/22/2019

```
34567 Robert Brooks   36 Pine St      Fall River MA 02720 04-APR-98        50 Amy Costa

6 rows selected.
```

I now when in and modified the code to put an ELSE clause in. When the v_yrgoal is greater than 250 I will increase the goal by 10% otherwise I will increase the goal by 5%. In looking at the code, notice that the processing that takes place in the IF is followed by a ; and the processing that takes place in the ELSE is followed by a ;. Also remember the structure which has the IF condition THEN and the terminating END IF;.

SQL CODE:

```
SQL> edit donor0a
```

PL/SQL CODE:

:= assignment

```
SET VERIFY OFF
DECLARE
   v_idno    VARCHAR2(5) := &input_idno;
   v_yrgoal  NUMBER(7,2);
BEGIN
   SELECT yrgoal INTO v_yrgoal
   FROM donornew
   WHERE idno = v_idno;
   IF v_yrgoal > 250 THEN
      v_yrgoal := v_yrgoal * 1.1;
   ELSE
      v_yrgoal := v_yrgoal * 1.05;
   END IF;
   UPDATE donornew
   SET yrgoal = v_yrgoal
   WHERE idno = v_idno;
   COMMIT;
   END;
/
SET VERIFY ON
```

I then verified the change by entering the idno for a person that had a goal less than >250 and the goal was upped by 5%.

SQL CODE:

```
SQL> @ donor0a
Enter value for input_idno: 34567

PL/SQL procedure successfully completed.

Input truncated to 13 characters

SQL> SELECT * FROM donornew;

IDNO  NAME            STADR           CITY       ST ZIP   DATEFST      YRGOAL CONTACT
----- --------------- --------------- ---------- -- ----- --------- --------- ------------
11111 Stephen Daniels 123 Elm St      Seekonk    MA 02345 03-JUL-98       500 John Smith
12121 Jennifer Ames   24 Benefit St   Providence RI 02045 24-MAY-97       440 Susan Jones
22222 Carl Hersey     24 Benefit St   Providence RI 02045 03-JAN-98           Susan Jones
23456 Susan Ash       21 Main St      Fall River MA 02720 04-MAR-92       120 Amy Costa
33333 Nancy Taylor    26 Oak St       Fall River MA 02720 04-MAR-92        50 John Adams
34567 Robert Brooks   36 Pine St      Fall River MA 02720 04-APR-98      52.5 Amy Costa
```

by the next IF. At the end, all of the IF statements are closed down with the END IF.

Two classic ways to do IF statements.

if { = T } else { = F }

if = T else = F end if

SQL CODE:

```
SQL> edit donor0b
```

PL/SQL CODE:

```
SET VERIFY OFF
DECLARE
   v_idno     VARCHAR2(5) := &input_idno;
   v_yrgoal   NUMBER(7,2);
BEGIN
   SELECT yrgoal INTO v_yrgoal
   FROM donornew
   WHERE idno = v_idno;
   IF v_yrgoal > 300 THEN
      v_yrgoal := v_yrgoal * 1.3;
   ELSE
      IF v_yrgoal > 200 THEN
         v_yrgoal := v_yrgoal * 1.2;
      ELSE
         IF v_yrgoal > 100 THEN
            v_yrgoal := v_yrgoal * 1.1;
         ELSE
            v_yrgoal := v_yrgoal * 1.05;
         END IF;
      END IF;
   END IF;
   UPDATE donornew
   SET yrgoal = v_yrgoal
   WHERE idno = v_idno;
   COMMIT;
   END;
/
SET VERIFY ON
```

To test this code I first put in idno 23456 which had a yrgoal that was > 100 and it was increased by 10%. Then I put in idno 11111 which had a yrgoal greater than 300 and it was increased by 30%.

SQL CODE:

```
SQL> @ donor0b
Enter value for input_idno: 23456

PL/SQL procedure successfully completed.

Input truncated to 13 characters
SQL> SELECT * FROM donornew;

IDNO  NAME            STADR           CITY       ST ZIP   DATEFST      YRGOAL CONTACT
----- --------------- --------------- ---------- -- ----- --------- --------- ------------
11111 Stephen Daniels 123 Elm St      Seekonk    MA 02345 03-JUL-98       500 John Smith
12121 Jennifer Ames   24 Benefit St   Providence RI 02045 24-MAY-97       440 Susan Jones
22222 Carl Hersey     24 Benefit St   Providence RI 02045 03-JAN-98           Susan Jones
23456 Susan Ash       21 Main St      Fall River MA 02720 04-MAR-92       132 Amy Costa
```

File Edit View History Bookmarks Tools Help

PL/SQL IF statements

www.pgrocer.net/Cis50/plsqlif1.html

```
33333 Nancy Taylor    26 Oak St       Fall River MA 02720 04-MAR-92        50 John Adams
34567 Robert Brooks   36 Pine St      Fall River MA 02720 04-APR-98      52.5 Amy Costa
```

Now I am going to test the IF..ELSIF which works like a case structure. IF the first condition is true it is executed. Otherwise the second condition is tested etc. The final ELSE is the catch-all for anything that did not meet one of the previous conditions. Notice that in this example, instead of assigning the yrgoal back to itself, I introduced a new variable called v_newgoal where I put the result of the calculation. This is simply another approach. If the yrgoal is less than a 100 it is increased by 10% and the if is exited. If not then yrgoal is checked to see if it is less than 250, if it is then it is increased by 20% and the if is exited. Anything else falls through to the else and is increased by 30%.

SQL CODE:

```
SQL> edit donor1
```

PL/SQL CODE:

If...ELSIF is a case structure

```
SET VERIFY OFF
DECLARE
   v_idno    VARCHAR2(5) :=&input_idno;
   v_yrgoal  NUMBER(7,2);
   v_newgoal NUMBER(7,2);
BEGIN
   SELECT yrgoal INTO v_yrgoal
   FROM donornew
   WHERE idno = v_idno;
   IF v_yrgoal < 100 THEN
          v_newgoal := v_yrgoal * 1.1;
     ELSIF v_yrgoal < 250 THEN
          v_newgoal := v_yrgoal * 1.2 ;
     ELSE
          v_newgoal := v_yrgoal * 1.3;
   END IF;
   UPDATE donornew
   SET yrgoal = v_newgoal
   WHERE idno = v_idno;
   COMMIT;
   END;
/
SET VERIFY ON
```

This code was tested with idno 34567 which should result in a 10 % increase and 23456 which should result in a 20% increase.

SQL CODE:

```
SQL> @ donor1
Enter value for input_idno: 34567

PL/SQL procedure successfully completed.

Input truncated to 13 characters
SQL> SELECT * FROM donornew;

IDNO  NAME            STADR           CITY       ST ZIP   DATEFST      YRGOAL CONTACT
----- --------------- --------------- ---------- -- ----- --------- --------- ------------
11111 Stephen Daniels 123 Elm St      Seekonk    MA 02345 03-JUL-98       650 John Smith
12121 Jennifer Ames   24 Benefit St   Providence RI 02045 24-MAY-97       440 Susan Jones
22222 Carl Hersey     24 Benefit St   Providence RI 02045 03-JAN-98           Susan Jones
```

Type here to search

10:09 AM 10/22/2019

File Edit View History Bookmarks Tools Help

PL/SQL IF statements

www.pgrocer.net/Cis50/plsqlif1.html

```
SET VERIFY OFF
DECLARE
   v_idno    VARCHAR2(5) := &input_idno;
   v_yrgoal  NUMBER(7,2);
BEGIN
   SELECT yrgoal INTO v_yrgoal
   FROM donornew
   WHERE idno = v_idno;
   IF v_yrgoal > 300 THEN
      v_yrgoal := v_yrgoal * 1.3;
   ELSE
      IF v_yrgoal > 200 THEN
         v_yrgoal := v_yrgoal * 1.2;
      ELSE
         IF v_yrgoal > 100 THEN
            v_yrgoal := v_yrgoal * 1.1;
         ELSE
            v_yrgoal := v_yrgoal * 1.05;
         END IF;
      END IF;
   END IF;
   UPDATE donornew
   SET yrgoal = v_yrgoal
   WHERE idno = v_idno;
   COMMIT;
   END;
/
SET VERIFY ON
```

To test this code I first put in idno 23456 which had a yrgoal that was > 100 and it was increased by 10%. Then I put in idno 11111 which had a yrgoal greater than 300 and it was increased by 30%.

SQL CODE:

```
SQL> @ donor0b
Enter value for input_idno: 23456

PL/SQL procedure successfully completed.

Input truncated to 13 characters
SQL> SELECT * FROM donornew;

IDNO  NAME            STADR           CITY       ST ZIP   DATEFST      YRGOAL CONTACT
----- --------------- --------------- ---------- -- ----- --------- --------- ------------
11111 Stephen Daniels 123 Elm St      Seekonk    MA 02345 03-JUL-98       500 John Smith
12121 Jennifer Ames   24 Benefit St   Providence RI 02045 24-MAY-97       440 Susan Jones
22222 Carl Hersey     24 Benefit St   Providence RI 02045 03-JAN-98           Susan Jones
23456 Susan Ash       21 Main St      Fall River MA 02720 04-MAR-92       132 Amy Costa
33333 Nancy Taylor    26 Oak St       Fall River MA 02720 04-MAR-92        50 John Adams
34567 Robert Brooks   36 Pine St      Fall River MA 02720 04-APR-98      52.5 Amy Costa

6 rows selected.

SQL> @ donor0b
Enter value for input_idno: 11111

PL/SQL procedure successfully completed.

Input truncated to 13 characters
```

Type here to search

10:10 AM
10/22/2019

Title: Oct 22-10:10 AM (12 of 29)

Now I am going to test the IF..ELSIF which works like a case structure. IF the first condition is true it is executed. Otherwise the second condition is tested etc. The final ELSE is the catch-all for anything that did not meet one of the previous conditions. Notice that in this example, instead of assigning the yrgoal back to itself, I introduced a new variable called v_newgoal where I put the result of the calculation. This is simply another approach. If the yrgoal is less than a 100 it is increased by 10% and the if is exited. If not then yrgoal is checked to see if it is less than 250, if it is then it is increased by 20% and the if is exited. Anything else falls through to the else and is increased by 30%.

SQL CODE:

```
SQL> edit donor1
```

PL/SQL CODE:

```
SET VERIFY OFF
DECLARE
   v_idno     VARCHAR2(5) :=&input_idno;
   v_yrgoal   NUMBER(7,2);
   v_newgoal  NUMBER(7,2);
BEGIN
   SELECT yrgoal INTO v_yrgoal
   FROM donornew
   WHERE idno = v_idno;
   IF v_yrgoal < 100 THEN
          v_newgoal := v_yrgoal * 1.1;
     ELSIF v_yrgoal < 250 THEN
          v_newgoal := v_yrgoal * 1.2 ;
     ELSE
          v_newgoal := v_yrgoal * 1.3;
   END IF;
   UPDATE donornew
   SET yrgoal = v_newgoal
   WHERE idno = v_idno;
   COMMIT;
   END;
/
SET VERIFY ON
```

This code was tested with idno 34567 which should result in a 10 % increase and 23456 which should result in a 20% increase.

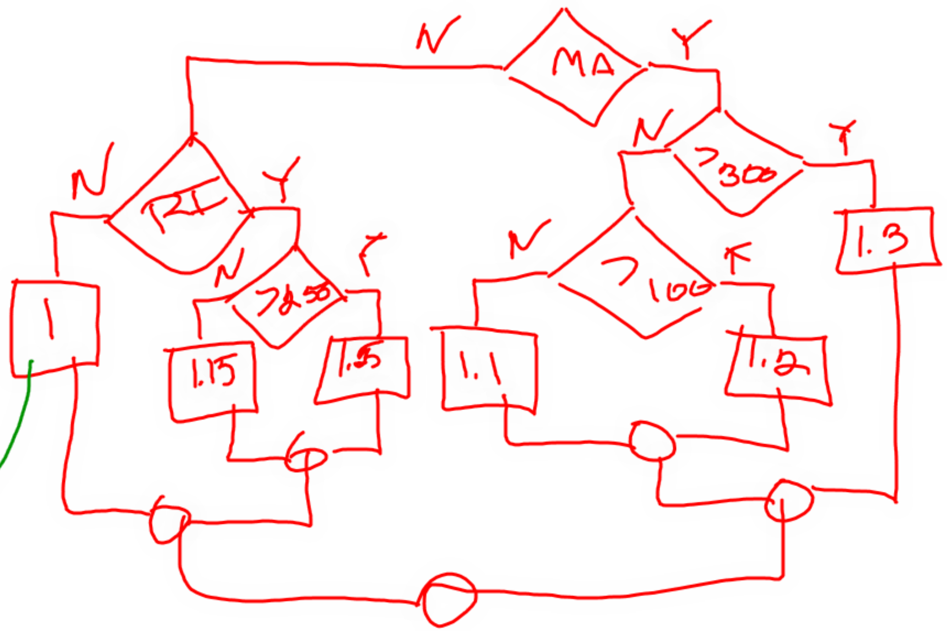
SQL CODE:

```
SQL> @ donor1
Enter value for input_idno: 34567

PL/SQL procedure successfully completed.

Input truncated to 13 characters
SQL> SELECT * FROM donornew;

IDNO  NAME            STADR           CITY       ST ZIP   DATEFST      YRGOAL CONTACT
----- --------------- --------------- ---------- -- ----- --------- --------- ------------
11111 Stephen Daniels 123 Elm St      Seekonk    MA 02345 03-JUL-98       650 John Smith
12121 Jennifer Ames   24 Benefit St   Providence RI 02045 24-MAY-97       440 Susan Jones
22222 Carl Hersey     24 Benefit St   Providence RI 02045 03-JAN-98           Susan Jones
23456 Susan Ash       21 Main St      Fall River MA 02720 04-MAR-92       132 Amy Costa
33333 Nancy Taylor    26 Oak St       Fall River MA 02720 04-MAR-92        50 John Adams
34567 Robert Brooks   36 Pine St      Fall River MA 02720 04-APR-98     57.75 Amy Costa
```

```
SQL> edit donor2
```

PL/SQL CODE:

```
SET VERIFY OFF
DECLARE
   v_idno    VARCHAR2(5) :=&input_idno;
   v_yrgoal  NUMBER(7,2);
   v_newgoal NUMBER(7,2);
   v_state   VARCHAR2(2);
BEGIN
   SELECT yrgoal, state INTO v_yrgoal, v_state
     FROM donornew
     WHERE idno = v_idno;
   IF v_state = 'MA' THEN
     IF v_yrgoal > 300 THEN
        v_newgoal := v_yrgoal * 1.3;
     ELSE
       IF v_yrgoal > 100 THEN
          v_newgoal := v_yrgoal * 1.2;
       ELSE
          v_newgoal := v_yrgoal * 1.1;
       END IF;
     END IF;
   ELSE
     IF v_state = 'RI' THEN
       IF v_yrgoal > 250 THEN
          v_newgoal := v_yrgoal * 1.25;
       ELSE
          v_newgoal := v_yrgoal * 1.15;
       END IF;
     ELSE
       v_newgoal := v_yrgoal;
     END IF;
   END IF;
   UPDATE donornew
     SET yrgoal = v_newgoal
     WHERE idno = v_idno;
   COMMIT;
   END;
/
SET VERIFY ON
```



SQL CODE:

```
SQL> @ donor2
Enter value for input_idno: 12121

PL/SQL procedure successfully completed.

Input truncated to 13 characters
SQL> SELECT * FROM donornew;

IDNO  NAME            STADR           CITY       ST ZIP   DATEFST      YRGOAL CONTACT
----- --------------- --------------- ---------- -- ----- --------- --------- ------------
11111 Stephen Daniels 123 Elm St      Seekonk    MA 02345 03-JUL-98       845 John Smith
12121 Jennifer Ames   24 Benefit St   Providence RI 02045 24-MAY-97       550 Susan Jones
22222 Carl Hersey     24 Benefit St   Providence RI 02045 03-JAN-98           Susan Jones
```

File Edit View History Bookmarks Tools Help

More IF statements with PL/SQL

www.pgrocer.net/Cis50/ormoreif.html

SQL CODE:

```
SQL> edit donor3
```

PL/SQL CODE:

```
SET VERIFY OFF
DECLARE
   v_idno    VARCHAR2(5) :=&input_idno;
   v_yrgoal  NUMBER(7,2);
   v_newgoal NUMBER(7,2);
   v_state   VARCHAR2(2);
BEGIN
   SELECT yrgoal, state INTO v_yrgoal, v_state
     FROM donornew
     WHERE idno = v_idno;
   IF v_yrgoal > 250 AND v_state = 'MA' THEN
      v_newgoal := v_yrgoal * 1.1 ;
   ELSE
      v_newgoal := v_yrgoal * 1.2;
   END IF;
   UPDATE donornew
     SET yrgoal = v_newgoal
     WHERE idno = v_idno;
END;
/
SET VERIFY ON
```

SQL CODE:

```
SQL> @ donor3
Enter value for input_idno: 23456

PL/SQL procedure successfully completed.

Input truncated to 13 characters
SQL> SELECT * FROM donornew;

IDNO  NAME            STADR           CITY       ST ZIP   DATEFST      YRGOAL CONTACT
----- --------------- --------------- ---------- -- ----- --------- --------- ------------
11111 Stephen Daniels 123 Elm St      Seekonk    MA 02345 03-JUL-98       845 John Smith
12121 Jennifer Ames   24 Benefit St   Providence RI 02045 24-MAY-97       550 Susan Jones
22222 Carl Hersey     24 Benefit St   Providence RI 02045 03-JAN-98           Susan Jones
23456 Susan Ash       21 Main St      Fall River MA 02720 04-MAR-92    190.08 Amy Costa
33333 Nancy Taylor    26 Oak St       Fall River MA 02720 04-MAR-92        50 John Adams
34567 Robert Brooks   36 Pine St      Fall River MA 02720 04-APR-98     63.53 Amy Costa

6 rows selected.
```

The code in donor3a has an embedded OR which asks if state = MA or yrgoal > 250. There is an else that handles all other conditions. The two examples below meet the criteria and are therefore processed with a 10% increase.

SQL CODE:

```
SQL> edit donor3a
```

Type here to search   10:18 AM 10/22/2019

Title: Oct 22-10:18 AM (15 of 29)

File Edit View History Bookmarks Tools Help

More IF statements with PL/SQL

www.pgrocer.net/Cis50/ormoreif.html

```
6 rows selected.
```

The code in donor3a has an embedded OR which asks if state = MA or yrgoal > 250. There is an else that handles all other conditions. The two examples below meet the criteria and are therefore processed with a 10% increase.

SQL CODE:

```
SQL> edit donor3a
```

PL/SQL CODE:

```
SET VERIFY OFF
DECLARE
   v_idno    VARCHAR2(5) :=&input_idno;
   v_yrgoal  NUMBER(7,2);
   v_newgoal NUMBER(7,2);
   v_state   VARCHAR2(2);
BEGIN
   SELECT yrgoal, state INTO v_yrgoal, v_state
     FROM donornew
     WHERE idno = v_idno;
   IF v_yrgoal > 250 OR v_state = 'MA' THEN
      v_newgoal := v_yrgoal * 1.1 ;
   ELSE
      v_newgoal := v_yrgoal * 1.2;
   END IF;
   UPDATE donornew
     SET yrgoal = v_newgoal
     WHERE idno = v_idno;
END;
/
SET VERIFY ON
```

SQL CODE:

```
SQL> @ donor3a

Enter value for input_idno: 12121

PL/SQL procedure successfully completed.

Input truncated to 13 characters
SQL> @ donor3a
Enter value for input_idno: 33333

PL/SQL procedure successfully completed.

Input truncated to 13 characters
SQL> SELECT * FROM donornew;

IDNO  NAME            STADR           CITY       ST ZIP   DATEFST      YRGOAL CONTACT
----- --------------- --------------- ---------- -- ----- --------- --------- ------------
11111 Stephen Daniels 123 Elm St      Seekonk    MA 02345 03-JUL-98       845 John Smith
12121 Jennifer Ames   24 Benefit St   Providence RI 02045 24-MAY-97       605 Susan Jones
22222 Carl Hersey     24 Benefit St   Providence RI 02045 03-JAN-98           Susan Jones
23456 Susan Ash       21 Main St      Fall River MA 02720 04-MAR-92    190.08 Amy Costa
33333 Nancy Taylor    26 Oak St       Fall River MA 02720 04-MAR-92        55 John Adams
```

Type here to search

10:18 AM
10/22/2019

Title: Oct 22-10:18 AM (16 of 29)

File Edit View History Bookmarks Tools Help

More IF statements with PL/SQL

www.pgrocer.net/Cis50/ormoreif.html

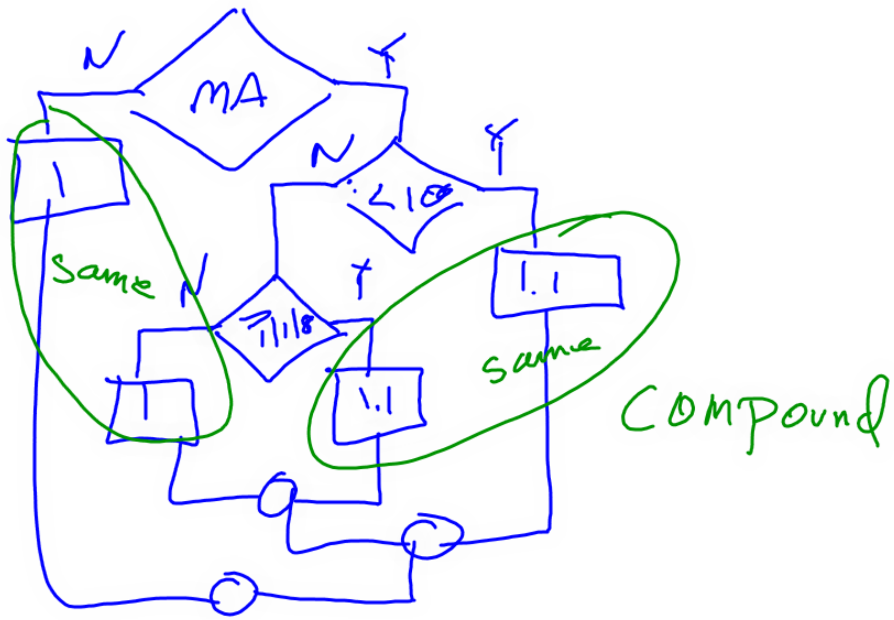
SQL CODE:

```
SQL> edit donor3b
```

PL/SQL CODE:

MA and either of 2 other conditions

```
SET VERIFY OFF
DECLARE
   v_idno    VARCHAR2(5) :=&input_idno;
   v_yrgoal  donornew.yrgoal%TYPE;
   v_newgoal donornew.yrgoal%TYPE;
   v_state   donornew.state%TYPE;
   v_datefst donornew.datefst%TYPE;
BEGIN
   SELECT yrgoal, state, datefst INTO v_yrgoal, v_state, v_datefst
     FROM donornew
     WHERE idno = v_idno;
   IF v_state = 'MA' AND (v_yrgoal < 100 OR v_datefst > '01-JAN-98') THEN
      v_newgoal := v_yrgoal * 1.1 ;
   ELSE
      v_newgoal := v_yrgoal;
   END IF;
   UPDATE donornew
     SET yrgoal = v_newgoal
     WHERE idno = v_idno;
END;
/
SET VERIFY ON
```



SQL CODE:

```
SQL> @ donor3b
Enter value for input_idno: 33333

PL/SQL procedure successfully completed.

Input truncated to 13 characters
SQL> @ donor3b
Enter value for input_idno: 12121

PL/SQL procedure successfully completed.

Input truncated to 13 characters
SQL> @ donor3b
Enter value for input_idno: 23456

PL/SQL procedure successfully completed.

Input truncated to 13 characters
SQL> SELECT * FROM donornew;

IDNO  NAME            STADR           CITY       ST ZIP   DATEFST       YRGOAL CONTACT
----- --------------- --------------- ---------- -- ----- --------- ---------- ------------
11111 Stephen Daniels 123 Elm St      Seekonk    MA 02345 03-JUL-98        845 John Smith
12121 Jennifer Ames   24 Benefit St   Providence RI 02045 24-MAY-97        605 Susan Jones
22222 Carl Hersey     24 Benefit St   Providence RI 02045 03-JAN-98            Susan Jones
```

Type here to search

10:23 AM
10/22/2019

SQL CODE:

```
SQL> edit donor3c
```

PL/SQL CODE:

AND resolves before OR

```
SET VERIFY OFF
DECLARE
   v_idno    VARCHAR2(5) :=&input_idno;
   v_yrgoal  donornew.yrgoal%TYPE;
   v_newgoal donornew.yrgoal%TYPE;
   v_state   donornew.state%TYPE;
   v_datefst donornew.datefst%TYPE;
BEGIN
   SELECT yrgoal, state, datefst INTO v_yrgoal, v_state, v_datefst
     FROM donornew
     WHERE idno = v_idno;
   IF v_state = 'MA' AND v_yrgoal < 100 OR v_datefst > '01-JAN-98' THEN
      v_newgoal := v_yrgoal * 1.1 ;
   ELSE
      v_newgoal := v_yrgoal;
   END IF;
   UPDATE donornew
     SET yrgoal = v_newgoal
     WHERE idno = v_idno;
END;
/
SET VERIFY ON
```

group

stands alone

SQL CODE:

```
SQL> @ donor3c
Enter value for input_idno: 12121

PL/SQL procedure successfully completed.

Input truncated to 13 characters
SQL> @ donor3c
Enter value for input_idno: 33333

PL/SQL procedure successfully completed.

Input truncated to 13 characters
SQL> SELECT * FROM donornew;

IDNO  NAME            STADR           CITY       ST ZIP   DATEFST      YRGOAL CONTACT
----- --------------- --------------- ---------- -- ----- --------- --------- ------------
11111 Stephen Daniels 123 Elm St      Seekonk    MA 02345 03-JUL-98       845 John Smith
12121 Jennifer Ames   24 Benefit St   Providence RI 02045 24-MAY-97       605 Susan Jones
22222 Carl Hersey     24 Benefit St   Providence RI 02045 03-JAN-98           Susan Jones
23456 Susan Ash       21 Main St      Fall River MA 02720 04-MAR-92    190.08 Amy Costa
33333 Nancy Taylor    26 Oak St       Fall River MA 02720 04-MAR-92     66.55 John Adams
34567 Robert Brooks   36 Pine St      Fall River MA 02720 04-APR-98     63.53 Amy Costa

6 rows selected.
```

Remember that you can check NULL operator with the IS NULL and that any expression that contains a NULL operator evaluates to NULL. In the example below if the state is RI and the yrgoal is either below 100 or NULL, the yrgoal gets reset to 100.

File Edit View History Bookmarks Tools Help

More IF statements with PL/SQL

www.pgrocer.net/Cis50/ormoreif.html

```
34567 Robert Brooks   36 Pine St      Fall River MA 02720 04-APR-98     63.53 Amy Costa

6 rows selected.
```

Remember that you can check NULL operator with the IS NULL and that any expression that contains a NULL operator evaluates to NULL. In the example below if the state is RI and the yrgoal is either below 100 or NULL, the yrgoal gets reset to 100. Record 22222 is from RI and has NULL as the yrgoal so the change gets made and yrgoal is reset to 100.

SQL CODE:

```
SQL> edit donor3d
```

PL/SQL CODE:

```
SET VERIFY OFF
DECLARE
   v_idno    VARCHAR2(5) :=&input_idno;
   v_yrgoal  donornew.yrgoal%TYPE;
   v_newgoal donornew.yrgoal%TYPE;
   v_state   donornew.state%TYPE;
BEGIN
   SELECT yrgoal, state INTO v_yrgoal, v_state
     FROM donornew
     WHERE idno = v_idno;
   IF v_state = 'RI' AND (v_yrgoal < 100 OR v_yrgoal IS NULL) THEN
      v_newgoal := 100;
   ELSE
      v_newgoal := v_yrgoal;
   END IF;
   UPDATE donornew
     SET yrgoal = v_newgoal
     WHERE idno = v_idno;
END;
/
SET VERIFY ON
```

SQL CODE:

```
SQL> @ donor3d
Enter value for input_idno: 22222

PL/SQL procedure successfully completed.

Input truncated to 13 characters
SQL> SELECT * FROM donornew;

IDNO  NAME            STADR           CITY       ST ZIP   DATEFST      YRGOAL CONTACT
----- --------------- --------------- ---------- -- ----- --------- --------- -----------
11111 Stephen Daniels 123 Elm St      Seekonk    MA 02345 03-JUL-98       845 John Smith
12121 Jennifer Ames   24 Benefit St   Providence RI 02045 24-MAY-97       605 Susan Jones
22222 Carl Hersey     24 Benefit St   Providence RI 02045 03-JAN-98       100 Susan Jones
23456 Susan Ash       21 Main St      Fall River MA 02720 04-MAR-92    190.08 Amy Costa
33333 Nancy Taylor    26 Oak St       Fall River MA 02720 04-MAR-92     66.55 John Adams
34567 Robert Brooks   36 Pine St      Fall River MA 02720 04-APR-98     63.53 Amy Costa

6 rows selected.
```

Type here to search

10:25 AM
10/22/2019

File Edit View History Bookmarks Tools Help

PL/SQL Loops

www.pgrocer.net/Cis50/orloops.html

incremented by 10 each pass through the loop. No data was put in the column data_in.

EXIT can be inserted multiple times in the code. From a programming point of view, I prefer EXIT once at the bottom of the loop to avoid confusion. Using EXIT to leave the loop before it is complete is a programming practice that I try to avoid.

PL/SQL CODE:

Starting loops

```
DECLARE
   v_recno    testloop.rec_no%TYPE :=1;
   v_ctr      testloop.ctr%TYPE :=100;
BEGIN
   LOOP
     INSERT INTO testloop(rec_no, ctr)
        VALUES(v_recno, v_ctr);
     v_recno :=v_recno + 1;
     v_ctr := v_ctr + 10;
     IF v_recno > 5 THEN
         EXIT;
     END IF;
   END LOOP;
END;
/
```

SQL CODE:

```
SQL> @ basicloop1
Input truncated to 1 characters

PL/SQL procedure successfully completed.
SQL> SELECT * FROM testloop;

   REC_NO       CTR DATA_IN
--------- --------- ----------
        1       100
        2       110
        3       120
        4       130
        5       140
```

Next I deleted all of the records from testloop so I start again.

SQL CODE:

```
SQL> DELETE FROM testloop;

5 rows deleted.

SQL> edit basicloop2
```

The change in this code is that I eliminated the IF statement to test for exiting the loop and instead used the EXIT WHEN with the condition. The condition is the same, when v_recno > 5. Again I prefer placing this test at the bottom of the loop. Note that when I code a loop, I always need to provide some means of exiting or the loop will run indefinitely.

PL/SQL CODE:

Type here to search

10:30 AM
10/22/2019

File Edit View History Bookmarks Tools Help

PL/SQL Loops

www.pgrocer.net/Cis50/orloops.html

Next I deleted all of the records from testloop so I start again.

SQL CODE:

```
SQL> DELETE FROM testloop;

5 rows deleted.

SQL> edit basicloop2
```

The change in this code is that I eliminated the IF statement to test for exiting the loop and instead used the EXIT WHEN with the condition. The condition is the same, when v_recno > 5. Again I prefer placing this test at the bottom of the loop. Note that when I code a loop, I always need to provide some means of exiting or the loop will run indefinitely.

PL/SQL CODE:

```
DECLARE
   v_recno    testloop.rec_no%TYPE :=1;
   v_ctr      testloop.ctr%TYPE :=100;
BEGIN
   LOOP
     INSERT INTO testloop(rec_no, ctr)
        VALUES(v_recno, v_ctr);
     v_recno :=v_recno + 1;
     v_ctr := v_ctr + 10;
     EXIT WHEN v_recno > 5;
   END LOOP;
END;
/
```

SQL CODE:

```
SQL> @ basicloop2
Input truncated to 1 characters

PL/SQL procedure successfully completed.

SQL> SELECT * FROM testloop;

   REC_NO       CTR DATA_IN
--------- --------- ----------
        1       100
        2       110
        3       120
        4       130
        5       140
```

In the example below, I first deleted all the records from testloop and then I went in and made a change to the code that set the initial value of v_recno to 6. Notice that in the basic loop structure, you have execution of the loop at least once even if the exit condition has been met prior to entering the loop. In other words, it enters the loop before testing. The insert could have been controlled by an IF so that no insert would be executed, but the loop would still be processed. This is illustrated in the second example below.

Type here to search

10:31 AM
10/22/2019

Title: Oct 22-10:31 AM (21 of 29)

File Edit View History Bookmarks Tools Help

PL/SQL Loops

www.pgrocer.net/Cis50/orloops.html

```
   REC_NO       CTR DATA_IN
--------- --------- ----------
        1       100
        2       110
        3       120
        4       130
        5       140
```

In the example below, I first deleted all the records from testloop and then I went in and made a change to the code that set the initial value of v_recno to 6. Notice that in the basic loop structure, you have execution of the loop at least once even if the exit condition has been met prior to entering the loop. In other words, it enters the loop before testing. The insert could have been controlled by an IF so that no insert would be executed, but the loop would still be processed. This is illustrated in the second example below.

SQL CODE:

```
SQL> edit basicloop3
```

PL/SQL CODE:

```
DECLARE
   v_recno    testloop.rec_no%TYPE :=6;
   v_ctr      testloop.ctr%TYPE :=100;
BEGIN
   LOOP
     INSERT INTO testloop(rec_no, ctr)
        VALUES(v_recno, v_ctr);
     v_recno :=v_recno + 1;
     v_ctr := v_ctr + 10;
     EXIT WHEN v_recno > 5;
   END LOOP;
END;
/
```

SQL CODE:

```
SQL> @ basicloop3
Input truncated to 1 characters

PL/SQL procedure successfully completed.

SQL> SELECT * FROM testloop;

   REC_NO       CTR DATA_IN
--------- --------- ----------
6 100

SQL> edit basicloop3a
```

PL/SQL CODE:

```
DECLARE
   v_recno    testloop.rec_no%TYPE :=6;
   v_ctr      testloop.ctr%TYPE :=100;
BEGIN
   LOOP
     IF v_recno < 6 THEN
```

Type here to search

10:32 AM
10/22/2019

File Edit View History Bookmarks Tools Help

PL/SQL Loops

www.pgrocer.net/Cis50/orloops.html

```
SQL> @ basicloop3
Input truncated to 1 characters

PL/SQL procedure successfully completed.

SQL> SELECT * FROM testloop;

   REC_NO       CTR DATA_IN
--------- --------- ----------
6 100

SQL> edit basicloop3a
```

PL/SQL CODE:

```
DECLARE
   v_recno    testloop.rec_no%TYPE :=6;
   v_ctr      testloop.ctr%TYPE :=100;
BEGIN
   LOOP
     IF v_recno < 6 THEN
       INSERT INTO testloop(rec_no, ctr)
         VALUES(v_recno, v_ctr);
     END IF;
     v_recno :=v_recno + 1;
     v_ctr := v_ctr + 10;
     EXIT WHEN v_recno > 5;
   END LOOP;
END;
/
```

SQL CODE:

```
SQL> @ basicloop3a
Input truncated to 1 characters

PL/SQL procedure successfully completed.

SQL> SELECT * FROM testloop;

no rows selected
```

WHILE LOOP

The WHILE loop allows the programmer to put a condition on the WHILE statement that starts the loop. The syntax is:

```
WHILE condition LOOP
       Processing code
END LOOP;
```

This format calls for the condition to be evaluated at the start of each iteration of the loop. If the condition is TRUE then the loop is processed, if the condition is FALSE then the loop is not processed. With loops it is important to have something to exit the loop and something that impacts or changes the condition on which you base that exit.

Type here to search

10:33 AM
10/22/2019

Title: Oct 22-10:33 AM (23 of 29)

File Edit View History Bookmarks Tools Help

PL/SQL Loops

www.pgrocer.net/Cis50/orloops.html

```
PL/SQL procedure successfully completed.

SQL> SELECT * FROM testloop;

no rows selected
```

WHILE LOOP

The WHILE loop allows the programmer to put a condition on the WHILE statement that starts the loop. The syntax is:

```
WHILE condition LOOP
     Processing code
END LOOP;
```

This format calls for the condition to be evaluated at the start of each iteration of the loop. If the condition is TRUE then the loop is processed, if the condition is FALSE then the loop is not processed. With loops it is important to have something to exit the loop and something that impacts or changes the condition on which you base that exit.

In the example below, the WHILE condition is v_recno < 6 which means the processing will continue until v_recno is no longer less than 6. Adding 1 to v_recno is the statement that impacts the condtion. When the add results in 6, the loop is stopped and so the insert for rec_no 6 never happens.

SQL CODE:

```
SQL> edit whileloop1
```

PL/SQL CODE:

```
DECLARE
   v_recno    testloop.rec_no%TYPE :=1;
   v_ctr      testloop.ctr%TYPE :=100;
BEGIN
   WHILE v_recno < 6 LOOP
     INSERT INTO testloop(rec_no, ctr)
        VALUES(v_recno, v_ctr);
     v_recno :=v_recno + 1;
     v_ctr := v_ctr + 10;
   END LOOP;
END;
/
```

SQL CODE:

```
SQL> @ whileloop1
Input truncated to 1 characters

PL/SQL procedure successfully completed.
SQL> SELECT * FROM testloop;

   REC_NO       CTR DATA_IN
--------- --------- ----------
        1       100
```

Type here to search

10:34 AM
10/22/2019

Title: Oct 22-10:34 AM (24 of 29)

File Edit View History Bookmarks Tools Help

PL/SQL Loops

www.pgrocer.net/Cis50/orloops.html

In the example below, the WHILE condition is v_recno < 6 which means the processing will continue until v_recno is no longer less than 6. Adding 1 to v_recno is the statement that impacts the condtion. When the add results in 6, the loop is stopped and so the insert for rec_no 6 never happens.

SQL CODE:

```
SQL> edit whileloop1
```

PL/SQL CODE:

```
DECLARE
   v_recno    testloop.rec_no%TYPE :=1;
   v_ctr      testloop.ctr%TYPE :=100;
BEGIN
   WHILE v_recno < 6 LOOP
     INSERT INTO testloop(rec_no, ctr)
        VALUES(v_recno, v_ctr);
     v_recno :=v_recno + 1;
     v_ctr := v_ctr + 10;
   END LOOP;
END;
/
```

SQL CODE:

```
SQL> @ whileloop1
Input truncated to 1 characters

PL/SQL procedure successfully completed.
SQL> SELECT * FROM testloop;

   REC_NO       CTR DATA_IN
--------- --------- ----------
        1       100
        2       110
        3       120
        4       130
        5       140
```

In the example below, I went in and changes the code so that v_recno had an initial value of 6. The WHILE condition prevented the loop from being entered and no output was produced.

SQL CODE:

```
SQL> edit whileloop2
```

PL/SQL CODE:

```
DECLARE
   v_recno    testloop.rec_no%TYPE :=6;
   v_ctr      testloop.ctr%TYPE :=100;
BEGIN
   WHILE v_recno < 6 LOOP
```

Type here to search

10:36 AM
10/22/2019

```
      5        140
```

In the example below, I went in and changes the code so that v_recno had an initial value of 6. The WHILE condition prevented the loop from being entered and no output was produced.

SQL CODE:

```
SQL> edit whileloop2
```

PL/SQL CODE:

```
DECLARE
   v_recno     testloop.rec_no%TYPE :=6;
   v_ctr       testloop.ctr%TYPE :=100;
BEGIN
   WHILE v_recno < 6 LOOP
     INSERT INTO testloop(rec_no, ctr)
        VALUES(v_recno, v_ctr);
     v_recno :=v_recno + 1;
     v_ctr := v_ctr + 10;
   END LOOP;
END;
/
```

Never enter loop

SQL CODE:

```
SQL> @ whileloop2
Input truncated to 1 characters

PL/SQL procedure successfully completed.

SQL> SELECT * FROM testloop;

no rows selected
```

FOR LOOP

The FOR loop controls the iterations in the FOR statement. The syntax is:

FOR index in [REVERSE] lower_bound..upper_bound LOOP
        Processing code
END LOOP;

The index is an implicitly declared integer which is automatically incremented by 1 on each iteration until the appropriate bound is reached. Notice that the index is not declared by the coder and can only be used in the loop specifications. Without the reverse clause, 1 is added starting from the lower_bound until the upper_bound is reached. With the reverse clause, everything is reverse and the upper_bound is decreased until the lower_bound is reached. The lower_bound and upper_bound can be literals or variable names or conditions that evaluate to an integer.

In the example below, the index is called I and the loop runs with the index starting at 1 and running through 5. The insert and incrementing of v_recno and v_ctr are strictly to supply data for the rows that are being inserted.

SQL CODE:

File Edit View History Bookmarks Tools Help

PL/SQL Loops

www.pgrocer.net/Cis50/orloops.html

FOR index in [REVERSE] lower_bound..upper_bound LOOP
Processing code
END LOOP;

The index is an implicitly declared integer which is automatically incremented by 1 on each iteration until the appropriate bound is reached. Notice that the index is not declared by the coder and can only be used in the loop specifications. Without the reverse clause, 1 is added starting from the lower_bound until the upper_bound is reached. With the reverse clause, everything is reverse and the upper_bound is decreased until the lower_bound is reached. The lower_bound and upper_bound can be literals or variable names or conditions that evaluate to an integer.

In the example below, the index is called I and the loop runs with the index starting at 1 and running through 5. The insert and incrementing of v_recno and v_ctr are strictly to supply data for the rows that are being inserted.

SQL CODE:

```
SQL> edit forloop1
```

PL/SQL CODE:

```
DECLARE
   v_recno    testloop.rec_no%TYPE :=1;
   v_ctr      testloop.ctr%TYPE :=100;
BEGIN
   FOR i IN 1..5 LOOP
     INSERT INTO testloop(rec_no, ctr)
        VALUES(v_recno, v_ctr);
     v_recno :=v_recno + 1;
     v_ctr := v_ctr + 10;
   END LOOP;
END;
/
```

SQL CODE:

```
SQL> @ forloop1
Input truncated to 1 characters

PL/SQL procedure successfully completed.

SQL> SELECT * FROM testloop;

   REC_NO       CTR DATA_IN
--------- --------- ----------
        1       100
        2       110
        3       120
        4       130
        5       140
```

The code below has been changed to use v_lower and v_upper to set the lower and upper limits of the FOR.

SQL CODE:

```
SQL> edit forloop2
```

Type here to search

10:38 AM
10/22/2019

```
Input truncated to 1 characters

PL/SQL procedure successfully completed.

SQL> SELECT * FROM testloop;

   REC_NO       CTR DATA_IN
--------- --------- ----------
        1       100
        2       110
        3       120
        4       130
        5       140
```

The code below has been changed to use v_lower and v_upper to set the lower and upper limits of the FOR.

SQL CODE:

```
SQL> edit forloop2
```

PL/SQL CODE:

```
DECLARE
   v_recno    testloop.rec_no%TYPE :=1;
   v_ctr      testloop.ctr%TYPE :=100;
   v_lower    NUMBER :=1;
   v_upper    NUMBER :=5;
BEGIN
   FOR i IN v_lower..v_upper LOOP
     INSERT INTO testloop(rec_no, ctr)
        VALUES(v_recno, v_ctr);
     v_recno :=v_recno + 1;
     v_ctr := v_ctr + 10;
   END LOOP;
END;
/
```

SQL CODE:

```
SQL> @ forloop2
Input truncated to 1 characters

PL/SQL procedure successfully completed.

SQL> SELECT * FROM testloop;

   REC_NO       CTR DATA_IN
--------- --------- ----------
        1       100
        2       110
        3       120
        4       130
        5       140
```

In this example I want to use the reverse and I decided to put the value of the index in as the record number. First, I eliminated the initial value on v_recno. Then I added in the REVERSE clause in the FOR loop and finally I moved the command that assigns a value to v_recno to before the insert and assigned the value of the index to it: v_recno := i;. This means that I will start with the value in v_upper which is 5 and immediately assign the 5 to the v_recno. It will then do the insert and increment v_ctr. Since

File Edit View History Bookmarks Tools Help

PL/SQL Loops

www.pgrocer.net/Cis50/orloops.html

a value to v_recno to before the insert and assigned the value of the index to it: v_recno := i;. This means that I will start with the value in v_upper which is 5 and immediately assign the 5 to the v_recno. It will then do the insert and increment v_ctr. Since the index has not reached the lower limit one will be subtracted from 5 and the index will now be 4. This is assigned to the v_recno and the next record will be inserted. Note that I could have eliminated v_recno completely and assigned the value of the index in the INSERT.

SQL CODE:

```
SQL> edit forloop3
```

PL/SQL CODE:

```
DECLARE
   v_recno    testloop.rec_no%TYPE;
   v_ctr      testloop.ctr%TYPE :=100;
   v_lower    NUMBER :=1;
   v_upper    NUMBER :=5;
BEGIN
   FOR i IN REVERSE v_lower..v_upper LOOP
     v_recno :=i;
     INSERT INTO testloop(rec_no, ctr)
        VALUES(v_recno, v_ctr);
     v_ctr := v_ctr + 10;
   END LOOP;
END;
/
```

Alternative PL/SQL CODE:

```
DECLARE
   v_ctr      testloop.ctr%TYPE :=100;
   v_lower    NUMBER :=1;
   v_upper    NUMBER :=5;
BEGIN
   FOR i IN REVERSE v_lower..v_upper LOOP
     INSERT INTO testloop(rec_no, ctr)
        VALUES(i, v_ctr);
     v_ctr := v_ctr + 10;
   END LOOP;
END;
/
SQL CODE:

SQL> @ forloop3
Input truncated to 1 characters

PL/SQL procedure successfully completed.

SQL> SELECT * FROM testloop;

   REC_NO       CTR DATA_IN
--------- --------- ----------
        5       100
        4       110
        3       120
        2       130
        1       140
```

Type here to search

10:40 AM
10/22/2019