PLSQL PROGRAMMING BASICS

PL/SQL is a procedural language that Oracle developed as an extension to standard SQL to provide a way to execute procedural logic on the database.

It, too, usually runs on the database server, but some Oracle products such as Developer/2000 also contain a PL/SQL engine that resides on the client. Thus, you can run your PL/SQL code on either the client or the server depending on which is more appropriate for the task at hand.

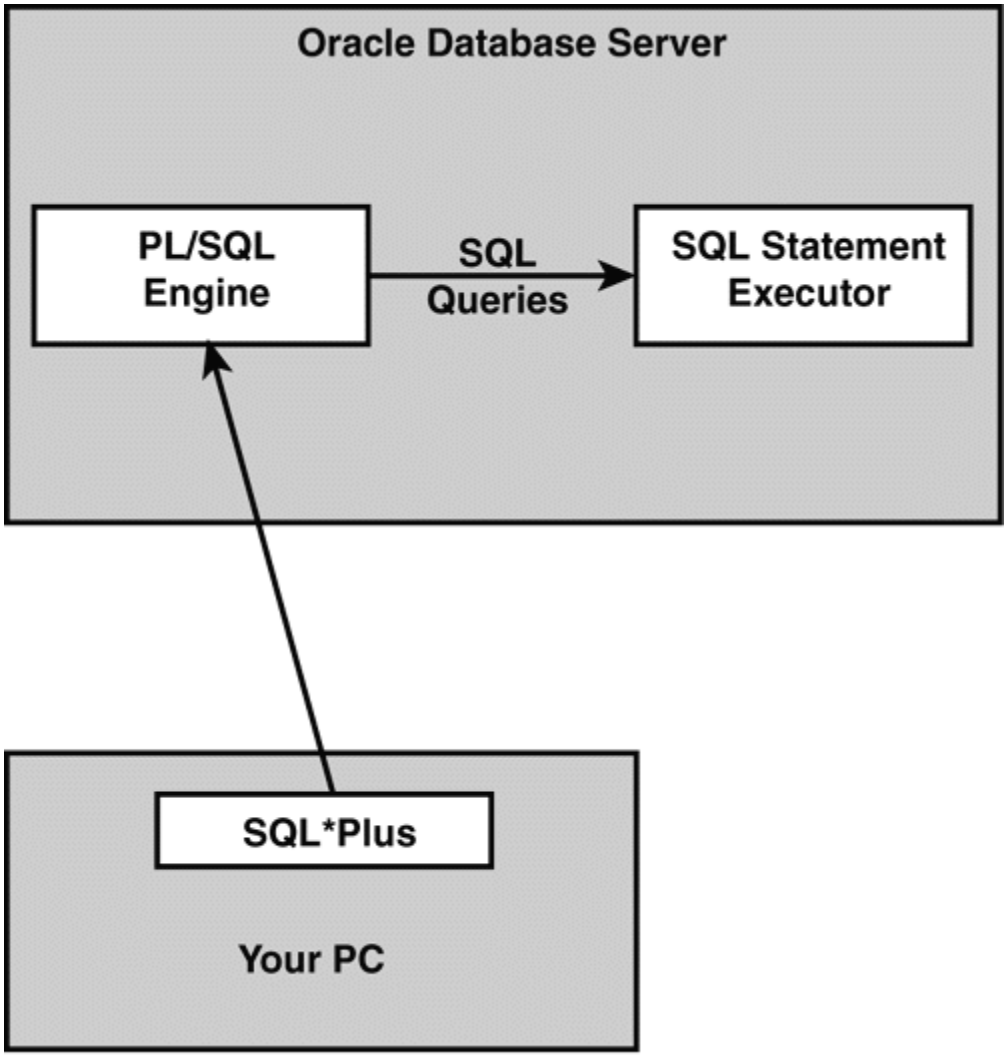
Unlike SQL, PL/SQL is procedural, not declarative. This means that your code specifies exactly how things get done.

As in SQL, however, you need some way to send your PL/SQL code up to the server for execution. PL/SQL also enables you to embed SQL statements within its procedural code.

This tight-knit relationship between PL/SQL, SQL, and SQL*Plus is the cause for some of the confusion between the products.

PL/SQL is executed in much the same manner. Type a PL/SQL block into SQL*Plus, and it is transmitted to the database server for execution. If there are any SQL statements in the PL/SQL code, they are sent to the server's SQL engine for execution, and the results are returned back to the PL/SQL program.

by dinesh


Oracle Database Server
PL/SQL Engine
SQL Queries
SQL Statement Executor
SQL*Plus
Your PC

by dinesh

PLSQL BLOCK STRUCTURE:

Declare

Begin

Exception

End;

- The **declare** section contains declaration of memory variables, constants, cursors etc.
- The **begin** section contains SQL executable statements and pl/SQL executable statements.
- The **exception** section contains code to handle errors that may arise during the execution of the code block.
- The **end** declares the end of pl/SQL block.

Display Message On Screen:

NOTE: To display messages to the user the **SERVEROUTPUT** should be set to **ON**.

SQL> SET SERVEROUTPUT ON

SERVEROUTPUT is a SQL*plus environment parameter that displays the information pased as a parameter to the PUT_LINE function.

TO produce or generate **OUTPUT** on screen following package:

DBMS_OUTPUT.PUT_LINE(‘ ’);

DBMS_OUTPUT : is a package that includes a number of procedure and functions that accumulate information in a buffer so that it can be retrieved later. These functions can also be used to display messages to the user.

The **dbms_output.put_line()** procedure takes exactly one argument and generates a line of text as output from the database server.
by dinesh

// Sample program for PLSQL block:

```
declare
str char(20);
begin
dbms_output.put_line('Hello World');
end;
```

Comments:

Single line : **--** (double lines)

```
begin
-- hello world
end;
```

Block : /* */

```
declare
a int :=10;
b int :=10;
begin
-- sum of numbers
a:=a+b;

/*
a:=a/10;
b:=b/10;
*/
end;
```

by dinesh

Initializing Variables:

Method 1:

```
declare
a int :=10; -- intializing at time of decleration
b int :=10;
begin
-- sum of numbers
a:=a+b;
end;
```

Method 2:

```
declare
a int;
b int;
begin
a:=10; -- intializing after decleration
b:=10;
-- sum of numbers
a:=a+b;
end;
```

User Inputs:

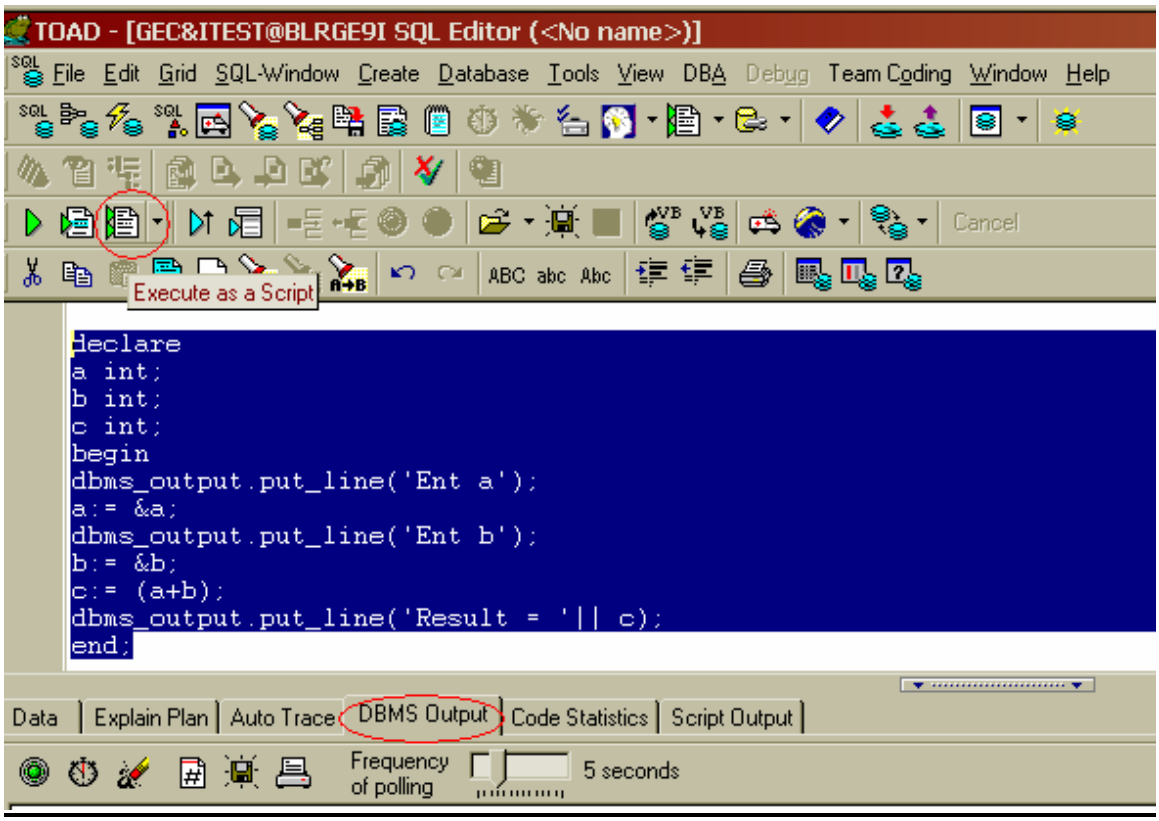
```
declare
a int;
b int;
c int;
begin
dbms_output.put_line('Ent a');
a:= &a;
dbms_output.put_line('Ent b');
b:= &b;
c:= (a+b);
dbms_output.put_line('Result = '|| c);
end;
```

by dinesh

Screen Shots:

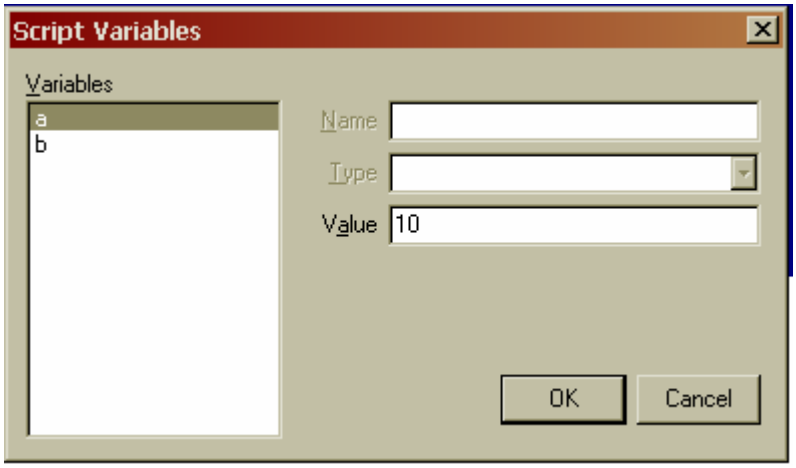
```
SQL> /
Enter value for a: 10
old   7: a:= &a;
new   7: a:= 10;
Enter value for b: 20
old   9: b:= &b;
new   9: b:= 20;
Ent a
Ent b
Result = 30

PL/SQL procedure successfully completed.

SQL> |
```

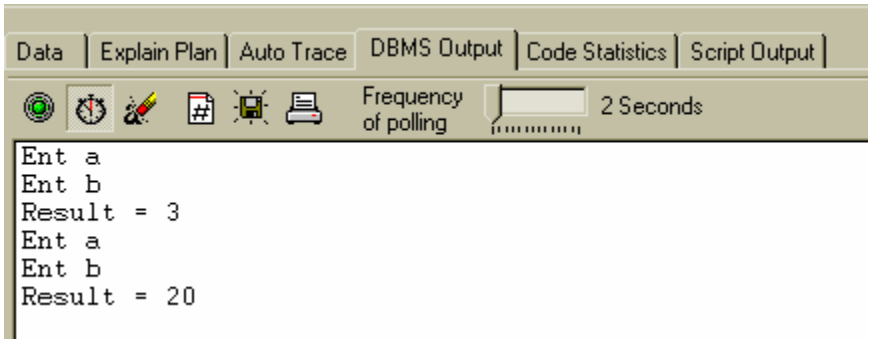
In TOAD:



To Execute the script press the icon show in the pic.
To view your output go to “DBMS Output”.

by dinesh


Script Variables
Variables
a
b
Name
Type
Value
10
OK
Cancel


Data
Explain Plan
Auto Trace
DBMS Output
Code Statistics
Script Output
Frequency of polling
2 Seconds
Ent a
Ent b
Result = 3
Ent a
Ent b
Result = 20

by dinesh

IF-THEN-ELSE Statement

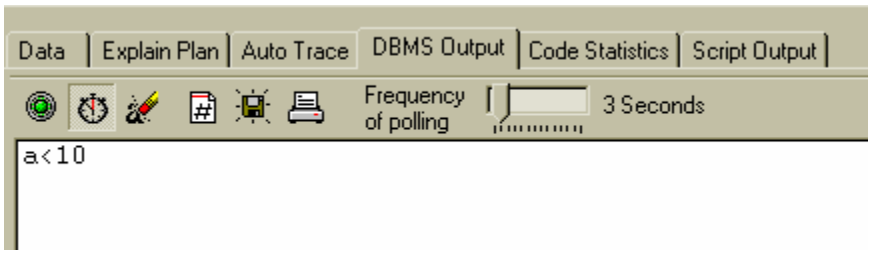
There are 3 different syntax for this statement:

Syntax 1:

If condition THEN
….
…..
End if;

```
declare
a int :=10;
begin
if a<10 then
   dbms_output.put_line('a<10');
end if;
end;
```

In TOAD:



Syntax 2:

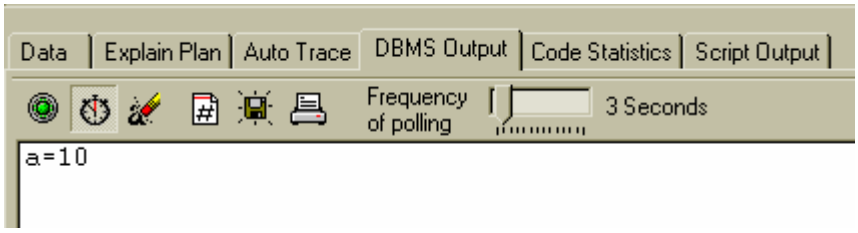
If condition THEN
….
Else
……
End if;

by dinesh

Syntax 3:

If condition THEN
….
Elsif condition THEN
….
Else
…..
End if;

```
declare
a int :=10;
begin
if a>10 then
   dbms_output.put_line('a>10');
elsif a=10 then
      dbms_output.put_line('a=10');
else
      dbms_output.put_line('a<10');
end if;
end;
```



by dinesh

CASE Statement:

Starting in Oracle 9i, you can use the **case** statement within an SQL statement. It has the functionality of an IF-THEN-ELSE statement.

Syntax:

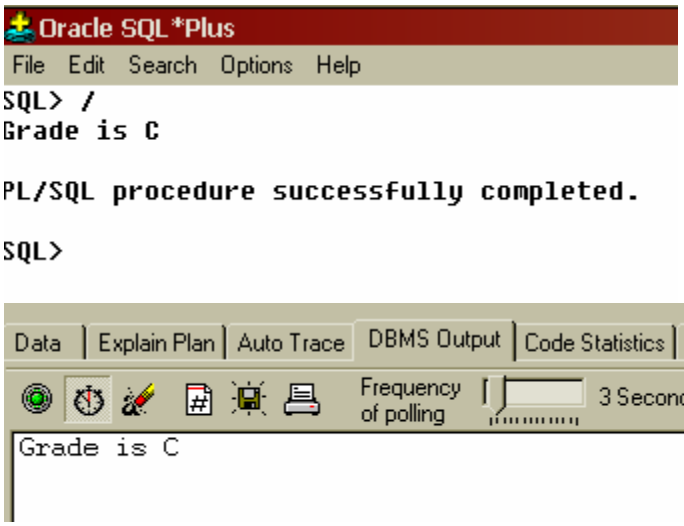
CASE <expression>

WHEN Condition_1 THEN
……
WHEN Condition_2 THEN
. ……
ELSE result;

END CASE;

// Program To perform case operation using local variable

```
declare
a int :=55;
begin
CASE
when (a <40) then
   dbms_output.put_line('Grade is fail');
when  (a>=40 and a<70) then
      dbms_output.put_line('Grade is C');
when (a>=70 and a<80) then
      dbms_output.put_line('Grade is B');
Else
      dbms_output.put_line('Grade is A');
end case;
end;
```



// Program To perform case operation in table

```
select case b
           when 40 then 'Grade is fail'
               when 70 then 'Grade is C'
               when 80 then 'Grade is B'
               Else 'Grade is A'
               end
              FROM num;
```

Data | Explain Plan | Auto Trace | DBMS Output | Code Statistics | Script Output

A	CASEBWHEN40THEN'GRADEISFAIL'WHEN70THEN'GRADEISC'WHEN80THEN'GRADEISB'ELSE'GRADEISA'END
10	Grade is A
20	Grade is A
3	Grade is A
3	Grade is A
33	Grade is A
55	Grade is A
2	Grade is A
100	Grade is A

```
SQL> /

CASEBWHEN40TH
-------------
Grade is A
Grade is A
Grade is A
Grade is A
Grade is A
Grade is A
Grade is A
Grade is A

8 rows selected.

SQL> |
```

by dinesh

Loop Statement:

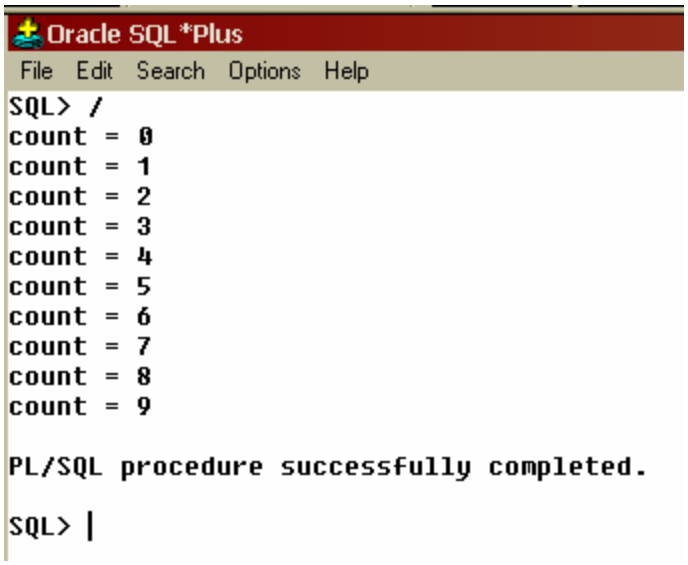
Syntax:

Loop
……..
……..
End Loop;

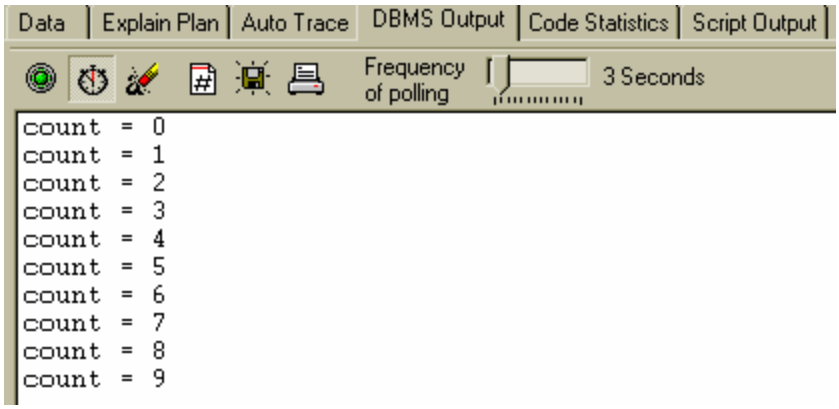
Note: This statement is mainly used when we are unaware, how many times we need to execute the loop.

This statement terminates when it finds EXIT or EXIT WHEN.

```
declare
cnt int :=0;
begin
       loop
       dbms_output.put_line('count = ' || cnt);
       cnt:=cnt+1;
       exit when cnt>=10;
       end loop;
 end;
```



by dinesh



FOR Loop:

Syntax 1:

FOR loop_counter IN Min_Count …. Max_Count
LOOP
……
……..
END LOOP;

Note: The counter will start at " Min " & ends at " Max ".

Syntax 2:

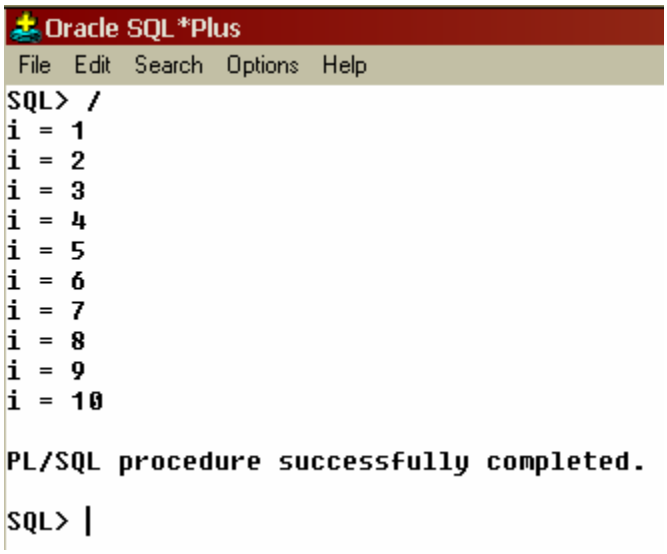
FOR loop_counter IN REVERSE Min_Count …. Max_Count
LOOP
……
……..
END LOOP;

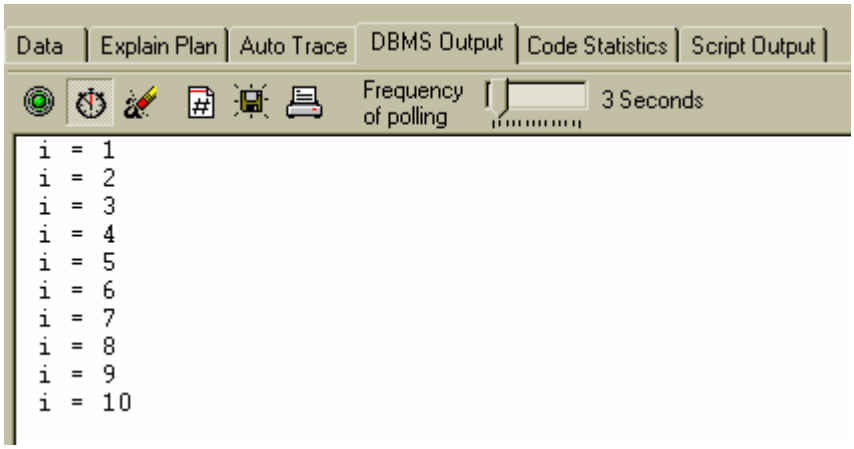
Note: The counter will start at " Max " & ends at " Min ".

by dinesh

Syntax 1:

```
declare
i int;
begin
        for i IN 1..10
        loop
        dbms_output.put_line(' i = '|| i);
        end loop;
end;
```





by dinesh

Syntax 2:

```
declare
i int;
begin
        for i IN REVERSE  1..10
        loop
        dbms_output.put_line(' i = '|| i);
        end loop;
end;
```

```
File  Edit  Search  Options  Help
SQL> /
i = 10
i = 9
i = 8
i = 7
i = 6
i = 5
i = 4
i = 3
i = 2
i = 1

PL/SQL procedure successfully completed.

SQL>
```

```
Data | Explain Plan | Auto Trace | DBMS Output | Code Statistics | Script Output
Frequency of polling  3 Seconds
i = 10
i = 9
i = 8
i = 7
i = 6
i = 5
i = 4
i = 3
i = 2
i = 1
```

by dinesh

CURSOR FOR Loop:

Syntax:

For Rec_index IN Cursor_name
Loop
.....
.....
End Loop;

```
Oracle SQL*Plus
File  Edit  Search  Options  Help
SQL> describe emp6;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER
 EMPNAME                                            VARCHAR2(20)
 EMPSALARY                                          NUMBER(8,2)
 GRADE                                              CHAR(1)
 EMPID                                              NUMBER(38)

SQL> select * from emp6;

     EMPNO EMPNAME               EMPSALARY G      EMPID
---------- -------------------- ---------- - ----------
         1 shovan                    22000 d
         2 dinesh                    30000 d
         3 senthil                   30000 d
         4 srikanth                  15000 d
         5 vishnu                    20000 d

SQL> |
```

```
create or replace function cur_loop(name emp6.empname%type) return
emp6.empsalary%type is
sum  emp6.empsalary%type;
cursor c1 is
select empsalary from emp6 where empname = name;
rec_index c1%ROWTYPE;
begin
      sum := 0;
       for rec_index in c1
       loop
        sum := rec_index.empsalary;
       end loop;

end cur_loop;
```

by dinesh

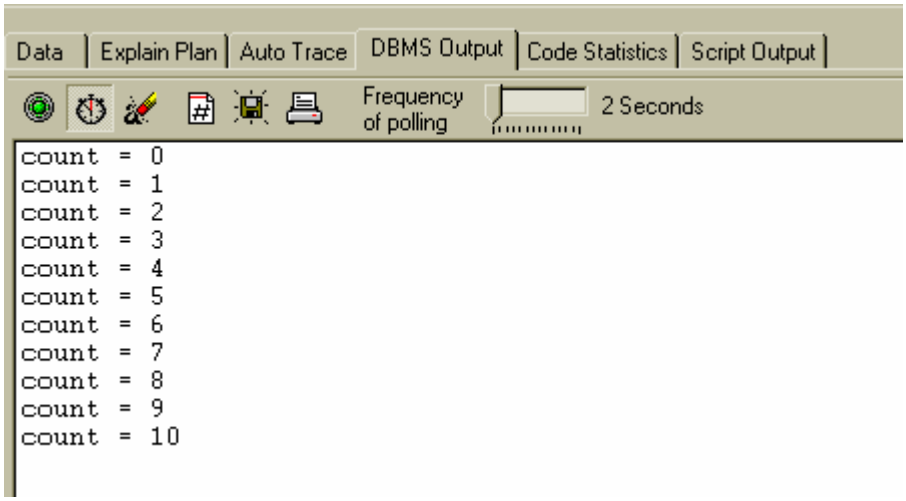
WHILE Loop:

Syntax:

WHILE Condition
Loop
……..
……..
End Loop;

This loop will be used when we are not sure about how many times to execute the body.

```
declare
cnt int :=0;

begin
       while cnt<=10
       loop
       dbms_output.put_line('count = '|| cnt);
       cnt:=cnt+1;
       end loop;

 end;
```



by dinesh

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> ed
Wrote file afiedt.buf

  1  declare
  2  cnt int :=0;
  3  begin
  4      while cnt<=10
  5      loop
  6      dbms_output.put_line('count = '|| cnt);
  7      cnt:=cnt+1;
  8      end loop;
  9*  end;
SQL> /
count = 0
count = 1
count = 2
count = 3
count = 4
count = 5
count = 6
count = 7
count = 8
count = 9
count = 10

PL/SQL procedure successfully completed.

SQL> |
```

by dinesh

EXCEPTION HANDLING:

System Exceptions:

Oracle has a standard set of exceptions already named as follows:

Oracle Exception Name	Oracle Error	Explanation
DUP_VAL_ON_INDEX	ORA-00001	You tried to execute an INSERT or UPDATE statement that has created a duplicate value in a field restricted by a unique index.
TIMEOUT_ON_RESOURCE	ORA-00051	You were waiting for a resource and you timed out.
TRANSACTION_BACKED_OUT	ORA-00061	The remote portion of a transaction has rolled back.
INVALID_CURSOR	ORA-01001	You tried to reference a cursor that does not yet exist. This may have happened because you've executed a FETCH cursor or CLOSE cursor before OPENing the cursor.
NOT_LOGGED_ON	ORA-01012	You tried to execute a call to Oracle before logging in.
LOGIN_DENIED	ORA-01017	You tried to log into Oracle with an invalid username/password combination.
NO_DATA_FOUND	ORA-01403	You tried one of the following: 1. You executed a SELECT INTO statement and no rows were returned. 2. You referenced an uninitialized row in a table. 3. You read past the end of file with the UTL_FILE package.
TOO_MANY_ROWS	ORA-01422	You tried to execute a SELECT INTO statement and more than one row was returned.
ZERO_DIVIDE	ORA-01476	You tried to divide a number by zero.
INVALID_NUMBER	ORA-01722	You tried to execute an SQL statement that tried to convert a string to a number, but it was unsuccessful.

by dinesh

STORAGE_ERROR	ORA-06500	You ran out of memory or memory was corrupted.
PROGRAM_ERROR	ORA-06501	This is a generic "Contact Oracle support" message because an internal problem was encountered.
VALUE_ERROR	ORA-06502	You tried to perform an operation and there was a error on a conversion, truncation, or invalid constraining of numeric or character data.
CURSOR_ALREADY_OPEN	ORA-06511	You tried to open a cursor that is already open.

USER Defined Exception:

Syntax:

CREATE OR REPLACE PROCEDURE procedure_name IS

Exception_name EXCEPTION;

BEGIN
……
……

EXCEPTION

WHEN condition THEN
……

WHEN OTHERS THEN
……..

END;

WHEN OTHER Clause:

The WHEN OTHERS clause is used to trap all remaining exceptions that have not been handled by your Named System Exceptions and Named Programmer-Defined Exceptions.

by dinesh

Table Used:

```
Oracle SQL*Plus
File  Edit  Search  Options  Help
SQL> DESCRIBE NUM;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 A                                                  NUMBER(38)
 B                                                  NUMBER(38)

SQL> SELECT * FROM NUM;

         A          B
---------- ----------
        10          4
        20         20
         3          1
         3         35
        33
        55          4
         2          1
       100        200

8 rows selected.

SQL>
```

```
create or replace procedure expn(x num.a%type, y num.b%type) is

check_exp exception;

begin

  if x>200 OR y>200 then
  RAISE check_exp;

  else
  insert into num values(x,y);
  end if;

  EXCEPTION

  when check_exp then
    raise_application_error(-10000,'X & Y should be <200');

  when OTHERS then
       raise_application_error(-10001,'application error');

end;
```

by dinesh

Running:

```
begin
expn(111,150);
end;
```

```
begin
expn(0,0);
end;
```

Output:

Data | Explain Plan | Auto Trace | DBMS Output | Code Statistics | Script Output

A	B
10	4
111	150
0	0
20	20
3	1
3	35
33	
55	4
2	1
100	200

```
begin
expn(270,300);
end;
```

Script Error

```
expn(270,300);
end;

ORA-21000: error number argument to raise_application_error of -10001
ORA-06512: at "GEC&ITEST.EXPN", line 17
ORA-06512: at line 2
```

Continue Script | Ignore All Errors | Terminate Script

by dinesh

Data | Explain Plan | Auto Trace | DBMS Output | Code Statistics | Script Output

```
begin
expn(270,300);
end;
ORA-21000: error number argument to raise_application_error of -10000 is out of
ORA-06512: at "GEC&ITEST.EXPN", line 17
ORA-06512: at line 2
```

Oracle9*i* Database Error Messages

http://download-uk.oracle.com/docs/cd/A97630_01/server.920/a96525/toc.htm

by dinesh