CURSORS

```
create table num
( a int,
  b int
 )

 insert into num values(1,2)

 insert into num values(3,1)

 insert into num values(22,12)

 insert into num values(50,100)

 insert into num values(10,9)
```

a	b
1	2
3	1
22	12
50	100
10	9

categorized cursors into the following topics:

Declare a Cursor

OPEN Statement

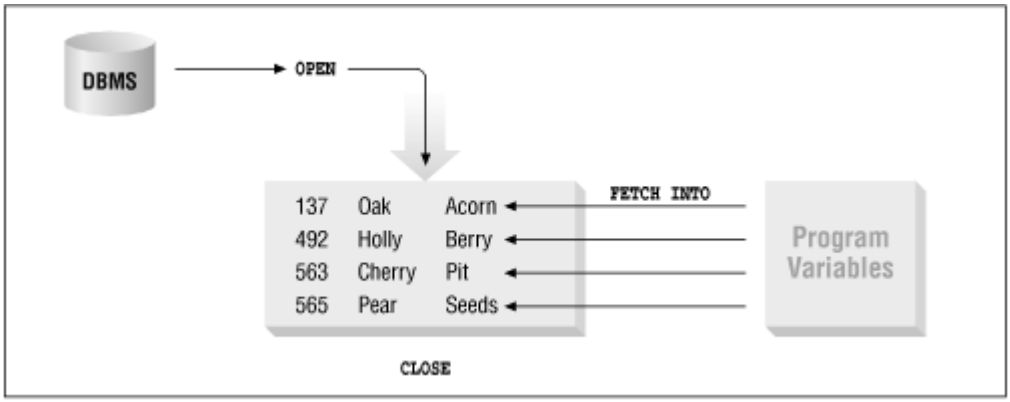
FETCH Statement

CLOSE Statement

Cursor Attributes (%FOUND, %NOTFOUND, etc)

SELECT FOR UPDATE Statement

WHERE CURRENT OF Statement



By dinesh

USING %TYPE

//simple program to select rows from the table using cursors

```
declare
                a1 num.a%type;
                b1 num.b%type;
                cursor c1 is
                select * from num;                          -- step 1
 Begin
            open c1;                                        -- step 2
                dbms_output.put_line('a' || '    ' || 'b');
                dbms_output.put_line('_______________');
                loop
                fetch c1 into a1,b1;
                exit when c1%NOTFOUND;                      --step 3
                dbms_output.put_line(a1 || '    ' || b1);
                end loop;
                close c1;                                   -- step 4
 end;
/
```

Note:

1. Here I have used *. So we need to declare temp_variables as in the table.
2. If we select 3 columns in SELECT query then we need to declare 3 temp_variable otherwise error will be raised in red line.

//Above program is implemented using selected columns in table.

```
declare
             a1 num.a%type;
            cursor c1 is
           select a from num;                               -- step 1
 Begin
             open c1;                                       -- step 2
                dbms_output.put_line('a');
                dbms_output.put_line('___');

        loop
                fetch c1 into a1;
                exit when c1%NOTFOUND;                      --step 3

        dbms_output.put_line(a1);
        end loop;
                close c1;                                   -- step 4
 end;/
```

By dinesh

// This program will select rows as per the condition and will display to the user

```
declare
                a1 num.a%type;
                b1 num.b%type;
                cursor c1 is
                select a,b from num where a<b;                    -- step 1
 Begin
          open c1;                                                -- step 2
                dbms_output.put_line('A' || '   ' || 'B');
                loop
                fetch c1 into a1,b1;
                exit when c1%NOTFOUND;                            --step 3
                dbms_output.put_line(a1  ||'    '|| b1);
                end loop;
                close c1;                                         -- step 4
 end;
/
```

//The above program is modified in such a way that when a>b the we need to delete the current row & to insert row as (b,a) i.e swapping

```
declare
                a1 num.a%type;
                b1 num.b%type;
                cursor c1 is
                select a,b from num where a<b for update;         -- step 1
 Begin
          open c1;                                                -- step 2
                dbms_output.put_line('A' || '   ' || 'B');
                loop
                fetch c1 into a1,b1;
                exit when c1%NOTFOUND;                            --step 3

                delete from num where current of c1;
                insert into  num values(b1,a1);

                end loop;

                close c1;                                         -- step 4
 end;
/
```

Note: SELECT stat can be as follows for this case:
SELECT * from num where a > b FOR UPDATE ;
SELECT * from num where a> b FOR UPDATE OF a;
By dinesh

USING %ROWTYPE

Note : while using %ROWTYPE no need to use **"Temp_ variables".**

// Sample program to implement this %Rowtype

```
declare
            cursor c1 is
        select * from num;                          -- step 1
            r_c1 c1%ROWTYPE;
Begin
         open c1;                                   -- step 2
            loop
                    fetch c1 into r_c1;
                    exit when c1%NOTFOUND;          --step 3
                    if r_c1.a > r_c1.b then
                    dbms_output.put_line(r_c1.a || '   is > than' || r_c1.b);
                    else
                    dbms_output.put_line(r_c1.b || '   is > than' || r_c1.a);
                    end if;
            end loop;
      close c1;                                     -- step 4
 end;
/
```

Note: in condition you are using two columns a & b. So in SELECT statement you need to put * or mention both columns.

//The same program is shown by using single column/attribute.

```
declare
            cursor c1 is
        select a from num;                          -- step 1
            r_c1 c1%ROWTYPE;
Begin
         open c1;                                   -- step 2
            loop
                    fetch c1 into r_c1;
                    exit when c1%NOTFOUND;          --step 3
                    if( r_c1.a > 35)  then
                    dbms_output.put_line(r_c1.a || '   is > than  ' || 35);
                    end if;
            end loop;
      close c1;                                     -- step 4
 end;
/
```

By dinesh

CURSOR ARRTIBUTES

Cursor Attributes	
Name	**Description**
%FOUND	Returns TRUE if record was fetched successfully, FALSE otherwise.
%NOTFOUND	Returns TRUE if record was not fetched successfully, FALSE otherwise.
%ROWCOUNT	Returns number of records fetched from cursor at that point in time.
%ISOPEN	Returns TRUE if cursor is open, FALSE otherwise.

```
//This program uses rowcount attribute.
declare
                cursor c1 is
            select a from num;                          -- step 1
                r_c1 c1%ROWTYPE;
Begin
             open c1;                                   -- step 2
                loop
                        fetch c1 into r_c1;
                        exit when c1%ROWCOUNT > 3 OR c1%NOTFOUND;  --step 3
                        if( r_c1.a > 35)  then
                        dbms_output.put_line(r_c1.a || '   is > than  ' || 35);
                        end if;
                end loop;
        close c1;                                       -- step 4
 end;
// The above Program using ISOPEN attribute
declare
                cursor c1 is
            select a from num;                          -- step 1
                r_c1 c1%ROWTYPE;
Begin
             if NOT c1%ISOPEN then
                open c1;                                -- step 2
                end if;
                loop
                        fetch c1 into r_c1;
                        exit when c1%ROWCOUNT > 3 OR c1%NOTFOUND;  --step 3
                        if( r_c1.a > 35)  then
                        dbms_output.put_line(r_c1.a || '   is > than  ' || 35);
```

By dinesh

```
                end if;
            end loop;
        close c1;                                   -- step 4
end;
```

USING WHILE structure To FETCH ROWS

```
            DECLARE
                a1 num.a%type;
                b1 num.b%type;
                CURSOR C1 IS
                SELECT * FROM num
                where a>3 and b>10;
            BEGIN
                OPEN C1;
                    FETCH C1 INTO a1,b1;
                WHILE C1%FOUND
                LOOP
                    FETCH C1 INTO a1,b1;
                dbms_output.put_line(a1 || '      '|| b1);
                END LOOP;
                CLOSE C1;
            END;
/
```

Note: two times we need to fetch the row for while loop one inside & one outside.

By dinesh

Working with REF CURSOR in PL/SQL

A REF CURSOR is basically a data type. A variable created based on such a data type is generally called a cursor variable. A cursor variable can be associated with different queries at run-time. The primary advantage of using cursor variables is their capability to pass result sets between sub programs (like stored procedures, functions, packages etc.).

USING %TYPE

```
1     declare
2     type r_cursor is REF CURSOR;
3     c1 r_cursor;
4     a1 num.a%type;

5     begin
6     open c1 for select a from num;

7     loop
8     fetch c1 into a1;
9     exit when c1%notfound;
10    dbms_output.put_line(a1);
11    end loop;

12    close c1;
13    end;
/
```

Explanation:

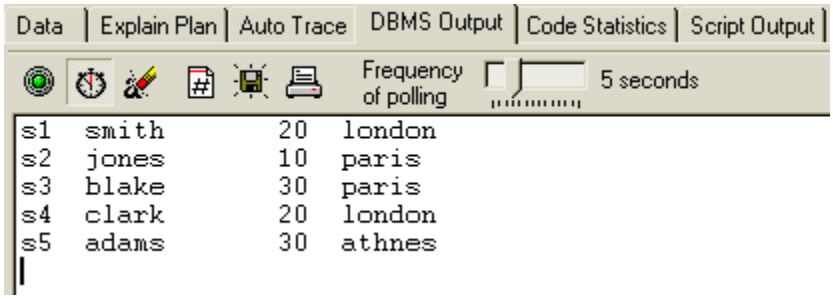
Step 2 is for defining a cursor r_cursor DATA TYPE which is of type REF CURSOR.
Step 3 Cursor variable is defind of type r_cursor
Step 6 Every cursor must open with associate SELECT statement.

By dinesh

USING %ROWTYPE

```
declare
 type rc1 is REF CURSOR;
 c1 rc1;

 r1 supplier%ROWTYPE;

begin
   if NOT c1%ISOPEN then
   open c1 for select * from supplier;
   end if;

   loop
   FETCH c1 into r1;
   exit WHEN c1%NOTFOUND;
   dbms_output.put_line(r1.s||'  '|| r1.sname||'  '||r1.status||'
'||r1.city);
   end loop;
  close c1;
end;
```

Output:



By dinesh

Table Used:

Column Name	Col ID	Pk	Data Type	Null?	Default
SHIPMENT_ID	1	1	INTEGER	N	
CUST_ID	2		INTEGER	Y	
WEIGHT	3		INTEGER	Y	
TRUCK_ID	4		INTEGER	Y	
DESTINATION	5		CHAR (20)	Y	
SHIP_DATE	6		DATE	Y	

Records in Table:

SHIPMENT_ID	CUST_ID	WEIGHT	TRUCK_ID	DESTINATION	SHIP_DATE
100	100	500	100	london	
101	101	100	102	paris	
102	101	300	103	london	
103	101	10	102	panamacity	12/12/2003
104	101	20	101	losangles	
105	102	200	102	rome	
106	100	50	101	siouxcity	9/18/2003
107	104	500	100	manhattan	
108	103	50	103	sanfransico	
109	104	25	101	sanfransico	
110	102	200	103	london	10/11/1998
111	103	100	101	london	9/9/1999
112	104	500	100	london	6/18/1988
113	104	200	100	london	10/11/1998
114	104	50	103	manhattan	9/9/1999
115	100	75	103	losangles	6/18/1988
116	101	55	102	baltimore	10/11/1998
117	103	45	101	paris	5/29/2003
118	103	45	100	rome	9/17/2002
119	103	45	102	losangles	7/1/2002
120	104	45	102	london	
121	100	150	102	siouxcity	
122	101	500	102	manhattan	
123	102	250	102	sanfransico	7/31/2002

By dinesh

Working with RECORD and REF CURSOR

Note:

%TYPE → Working with **one** record.

%ROWTYPE → Working with **multiple** record.

To create our own data type and specific number of values, We are going to use **TYPE & RECORD.**

Note:2

Now a question arises why we are using ref cursor?
We have one answer creating own data type. YES that's right.
But one more reason is also there.

Consider a scenario. When you use **ROWTYPE** you are selecting all the columns i.e. one record completely. Now you want to select only specific columns from one record and whenever you want to use those specific columns anywhere in PLSQL block we have to use our own record & own data type.

Tips:
consider we are using **VIEW instead of table.**(for understanding purpose only, this is not the concept.)
E.g.:

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

SHIPMENT_ID	CUST_ID	WEIGHT	TRUCK_ID	DESTINATION	SHIP_DATE
100	100	500	100	london	
101	101	100	102	paris	
102	101	300	103	london	
103	101	10	102	panamacity	12/12/2003
104	101	20	101	losangles	
105	102	200	102	rome	
106	100	50	101	siouxcity	9/18/2003
107	104	500	100	manhattan	
108	103	50	103	sanfransico	
109	104	25	101	sanfransico	
110	102	200	103	london	10/11/1998
111	103	100	101	london	9/9/1999
112	104	500	100	london	6/18/1988
113	104	200	100	london	10/11/1998
114	104	50	103	manhattan	9/9/1999

By dinesh

Now you want to work only with two columns shipment id and destination and consider this two columns as one complete record.

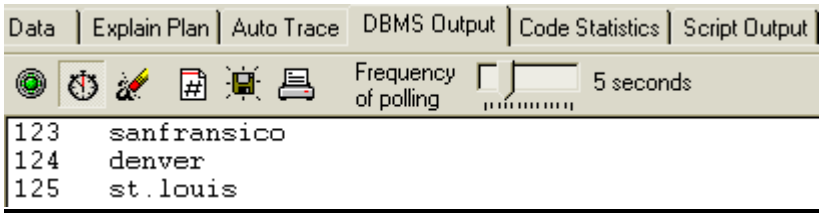
SHIPMENT_ID	DESTINATION
100	london
101	paris
102	london
103	panamacity
104	losangles
105	rome
106	siouxcity
107	manhattan
108	sanfransico

Let us consider the following example:

```
declare

type rc2 is REF CURSOR;
c2 rc2;

type rec1 is record
( sid int,
  dest char(20)
);

shipment_1 rec1;
begin
    open c2 for select shipment_id,destination from shipment2;

      loop
      fetch c2 into shipment_1;
      exit when c2%NOTFOUND;
      dbms_output.put_line(shipment_1.sid || '    ' || shipment_1.dest);
      end loop;
      close c2;

end;
```

By dinesh

Output:



Explanations:

```
type rec1 is record
( sid int,
  dest char(20)
);
```

The above defines a new data type named "shipment_1" (just like %ROWTYPE with limited specified fields) which can hold two fields, namely "sid" and "dest."

```
shipment_1 rec1;
```

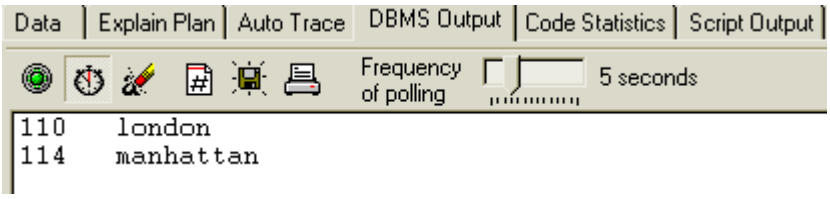
The above statement declares a variable "shipment_1er" based on the data type "rec1." This means that "shipment_1" internally contains the fields "sid" and "dest."

By dinesh

Working with more than one query with the same REF CURSOR

```
declare

type rc2 is REF CURSOR;
c2 rc2;

type rec1 is record
( sid int,
  dest char(20)
);

shipment_1 rec1;
begin
-- 1st query
    open c2 for select shipment_id,destination from shipment2 where
    shipment_id = 110;

      loop
      fetch c2 into shipment_1;
      exit when c2%NOTFOUND;
      dbms_output.put_line(shipment_1.sid || '    ' || shipment_1.dest);
      end loop;
      close c2;

-- 2nd query
   open c2 for select shipment_id,destination from shipment2 where
   shipment_id = 114;

      loop
      fetch c2 into shipment_1;
      exit when c2%NOTFOUND;
      dbms_output.put_line(shipment_1.sid || '    ' || shipment_1.dest);
      end loop;
      close c2;

end;
```

Output :



By dinesh

Working with REF CURSOR inside loops

```
declare

type rc2 is REF CURSOR;
c2 rc2;

type rec1 is record
( cid int,
  dest char(20)
);

rec_ship rec1;
begin
       for i in (select shipment_id, ship_date from shipment2) -- any
columns but only 2
       loop
             open c2 for select cust_id,destination from shipment2
             where shipment_id = i.shipment_id;
             dbms_output.put_line(i.ship_date);
             dbms_output.put_line('__________');
             loop
                   fetch c2 into rec_ship;
                   exit when c2%NOTFOUND;
                   dbms_output.put_line(rec_ship.cid || '    '||
                                        rec_ship.dest );
             end loop;
            close c2;z`
       end loop;
end;
```

Explanation:

```
for i in (select shipment_id, ship_date from shipment2)
 loop
    .
    .
end loop;
```

The above loop iterates continuously for each row of the "shipment2" table. The details of each row in "shipment2t" (like `shipment_id, ship_date` etc.) will be available in the variable "i." Using that variable (as part of the SELECT statement), then working with REF CURSOR as follows:

```
 open c2 for select cust_id,destination from shipment2
             where shipment_id = i.shipment_id;
```

By dinesh

Output:

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

Frequency of polling 5 seconds

```
___________
100    london

___________
101    paris

___________
101    london
12-DEC-03
___________
101    panamacity

___________
101    losangles

___________
102    rome
18-SEP-03
___________
100    siouxcity

___________
104    manhattan

___________
103    sanfransico

___________
104    sanfransico
11-OCT-98
```

By dinesh

Dealing with REF CURSOR in the sub-programs of a PL/SQL block

For easy understanding consider the above PLSQL code,

```
declare

type rc2 is REF CURSOR;
c2 rc2;

type rec1 is record
( cid int,
  dest char(20)
);

rec_ship rec1;
begin
       for i in (select shipment_id, ship_date from shipment2) -- any
columns but only 2
       loop
             open c2 for select cust_id,destination from shipment2
             where shipment_id = i.shipment_id;
             dbms_output.put_line(i.ship_date);
             dbms_output.put_line('__________');
             loop
                   fetch c2 into rec_ship;
                   exit when c2%NOTFOUND;
                   dbms_output.put_line(rec_ship.cid || '
                                        '|| rec_ship.dest );
             end loop;
            close c2;z`
       end loop;
end;
```

Note:

To use ref cursor in sub programs, put the highlighted code into the **procedure**. See the code below:

By dinesh

```
declare

type rc2 is REF CURSOR;
c2 rc2;

type rec1 is record
( cid int,
  dest char(20)
);

rec_ship rec1;

       procedure p_ref is
       begin
              loop
              fetch c2 into rec_ship;
              exit when c2%NOTFOUND;
              dbms_output.put_line(rec_ship.dest);
              end loop;
       end;

begin
       for i in (select shipment_id, ship_date from shipment2) -- any
columns but only 2
       loop
             open c2 for select cust_id,destination from shipment2
                         where shipment_id = i.shipment_id;
             dbms_output.put_line(i.ship_date);
             dbms_output.put_line('__________');

             p_ref;

       end loop;

end;
```

Explanation:

```
for i in (select shipment_id, ship_date from shipment2)
 loop
    .
    .
   p_ref;
    .
    .
 end loop;
```

According to the above loop, the sub-routine gets executed for every iteration, which displays the employee information for the respective department.

By dinesh

Output :

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

Frequency of polling 5 seconds

```
sanfransico
11-OCT-98

_____________
london
09-SEP-99

_____________
london
18-JUN-88

_____________
london
11-OCT-98

_____________
london
09-SEP-99

_____________
manhattan
18-JUN-88

_____________
losangles
11-OCT-98

_____________
baltimore
29-MAY-03

_____________
paris
17-SEP-02

_____________
rome
01-JUL-02

_____________
losangles
```

By dinesh

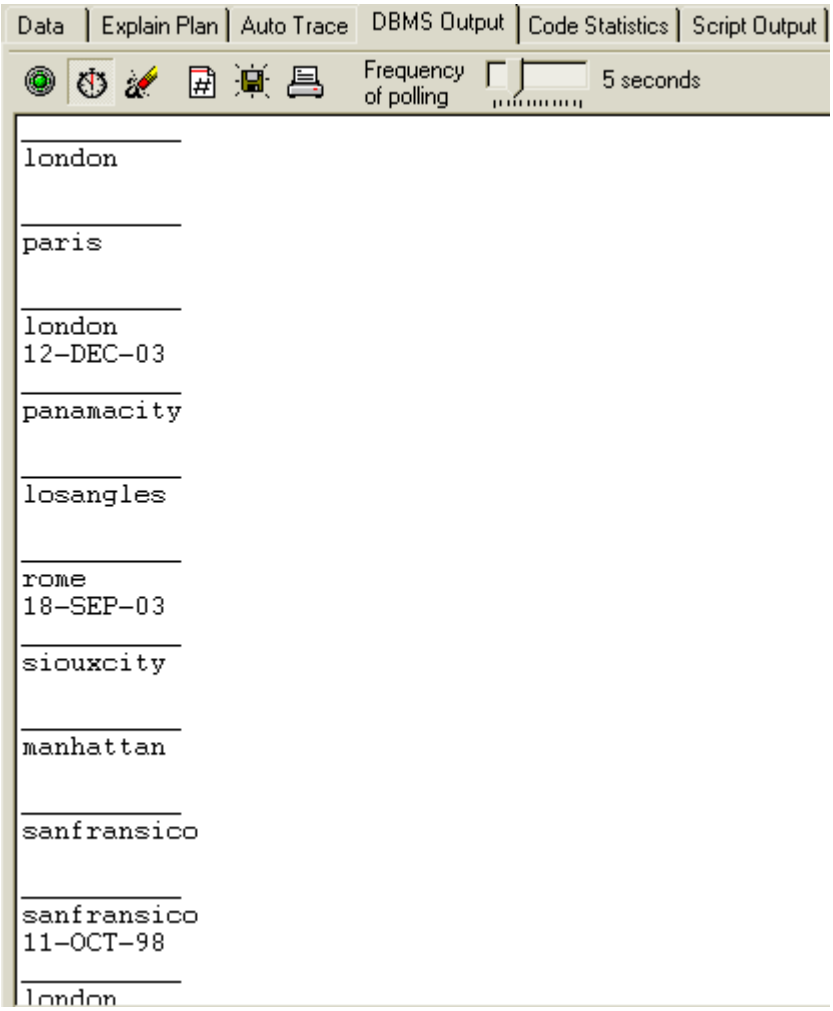
Passing REF CURSOR as parameters to sub-programs

- Every sub-program (or sub-routine) can accept values passed to it in the form of "parameters" (or arguments).
- Every parameter is very similar to a variable, but gets declared as part of a sub-program.

```
declare

type rc2 is REF CURSOR;
c2 rc2;

type rec1 is record
( cid int,
  dest char(20)
);

       procedure p_ref(pc2 rc2) is
   (or) procedure p_ref ( pc2 c2%TYPE )

       rec_ship rec1;

       begin
              loop
              fetch c2 into rec_ship;
              exit when c2%NOTFOUND;
              dbms_output.put_line(rec_ship.dest);
              end loop;
       end;

begin
       for i in (select shipment_id, ship_date from shipment2) -- any
columns but only 2
       loop
             open c2 for select cust_id,destination from shipment2 where
shipment_id = i.shipment_id;
           dbms_output.put_line(i.ship_date);
             dbms_output.put_line('__________');

             p_ref(c2);

       end loop;

   end;
```

By dinesh

Output :



By dinesh