CURSORS

cursors

ORACLE OPENS UNNAMED WORK AREAS (MEMORY BLOCKS) CALLED “PRIVATE SQL AREAS” THAT HOLDS THE RESULTS OF SELECT/DML STATEMENTS .

A CURSOR IS A PL/SQL CONSTRUCT THAT ALLOWS TO NAME THESE WORK AREAS SO AS TO ACCESS THE INFORMATION STORED IN IT.

CURSORS ARE OF TWO TYPES:

1. IMPLICIT CURSORS
2. EXPLICIT CURSORS

Cursors

IMPLICIT CURSORS:

ORACLE IMPLICITLY OPENS ,PROCESSES AND CLOSES THE CURSOR FOR ALL DML STATEMENTS(INSERT/DELETE/UPDATE) AS WELL AS FOR QUERIES THAT RETURN SINGLE ROW VALUES.

EXPLICIT CURSORS:

FOR SELECT STATEMENTS THAT RETURN MORE THAN ONE ROW WITHIN A PL/SQL BLOCK, WE MUST EXPLICITLY OPEN THE CURSORS.

Implicit Cursors

Implicit cursors are implicitly opened and closed by oracle. But, we can know the status of the most recently executed SQL statement within the PL/SQL block by looking into cursor attributes.

CURSOR ATTRIBUTES CONTAIN INFORMATION ABOUT THE MOST RECENTLY EXECUTED SQL STATEMENT.

There are 4 cursor attributes:

1. %ISOPEN
2. %FOUND
3. %NOTFOUND
4. %ROWCOUNT

USAGE OF A CURSOR ATTRIBUTE :

CURSOR_NAME%CURSOR_ATTRIBUTE

Implicit Cursors

THE NAME OF AN IMPLICIT CURSOR IS SQL.

Usage: SQL%CURSOR_ATTRIBUTE

SQL%ROWCOUNT	**Number of records affected by the most recent SQL statement**
SQL%FOUND	**Evaluates to TRUE if the most recent SQL statement affects one or more rows**
SQL%NOTFOUND	**Evaluates to TRUE if the most recent SQL statement does not affect any rows**
SQL%ISOPEN	**Always evaluates to FALSE because PL/SQL closes implicit cursors immediately after they are executed**

Implicit Cursors

Ex.

```
DECLARE
        v_num integer;
BEGIN
        UPDATE EMP
        SET SAL = SAL* 1.10
        WHERE DEPTNO= &DNO;
        v_num := SQL%ROWCOUNT;
        DBMS_OUTPUT.PUT_LINE(v_num || ‘ Rows  updated');
END;
```

Implicit Cursors

```
BEGIN
 DELETE FROM EMP
 WHERE HIREDATE < TO_DATE('31/12/1985','DD/MM/YYYY');
 IF SQL%NOTFOUND THEN
      DBMS_OUTPUT.PUT_LINE('NO EMPLOYEES IN THE TABLE');
 ELSIF SQL%ROWCOUNT > 10 THEN
      DBMS_OUTPUT.PUT_LINE ('More than 10 empoyees');
      COMMIT;
 ELSE
     DBMS_OUTPUT.PUT_LINE ('Less than 10 empoyees');
     ROLLBACK;
 END IF;
END;
```

Implicit Cursors

```
DECLARE
 V_SAL  EMP.SAL%TYPE;
 BEGIN
        SELECT NVL(SAL,0) INTO V_SAL
        FROM EMP
        WHERE EMPNO = &ENO;
        DBMS_OUTPUT.PUT_LINE ('Salary of the employee:  ' || v_sal);
         IF SQL%NOTFOUND THEN
           DBMS_OUTPUT.PUT_LINE('INVALID EMPNO');
         END IF;
 END;
```

If user enters invalid empno, SQL%NOTFOUND is not tested. Instead error propagates to calling environment. Why?

Explicit Cursors

There are 3 types of explicit cursors :

1. EXPLICIT CURSORS
2. CURSOR FOR LOOPS
3. CURSOR VARIABLES

Attribute	Type	Description
%ISOPEN	Boolean	Evaluates to TRUE if the cursor is open
%NOTFOUND	Boolean	Evaluates to TRUE if the most recent fetch does not return a row
%FOUND	Boolean	Evaluates to TRUE if the most recent fetch returns a row; complement of %NOTFOUND
%ROWCOUNT	Number	Evaluates to the total number of rows returned so far

Explicit Cursors

EXPLICIT CURSORS

1. DECLARING A CURSOR :

```
CURSOR CURSOR _NAME[(PARAMETERS)]
IS
SELECT STATEMENT;
```

2. OPENING THE CURSOR :

```
OPEN CURSOR_NAME[(PARAMETERS)];
```

Explicit Cursors

3. FETCHING THE CONTENTS OF CURSOR AREA (ACTIVE SET):

```
            FETCH CURSOR_NAME INTO VARIABLE(S);
```

NOTE: AFTER EACH FETCH THE CURSOR ADVANCES TO THE NEXT ROW OF THE ACTIVE SET.

EX.

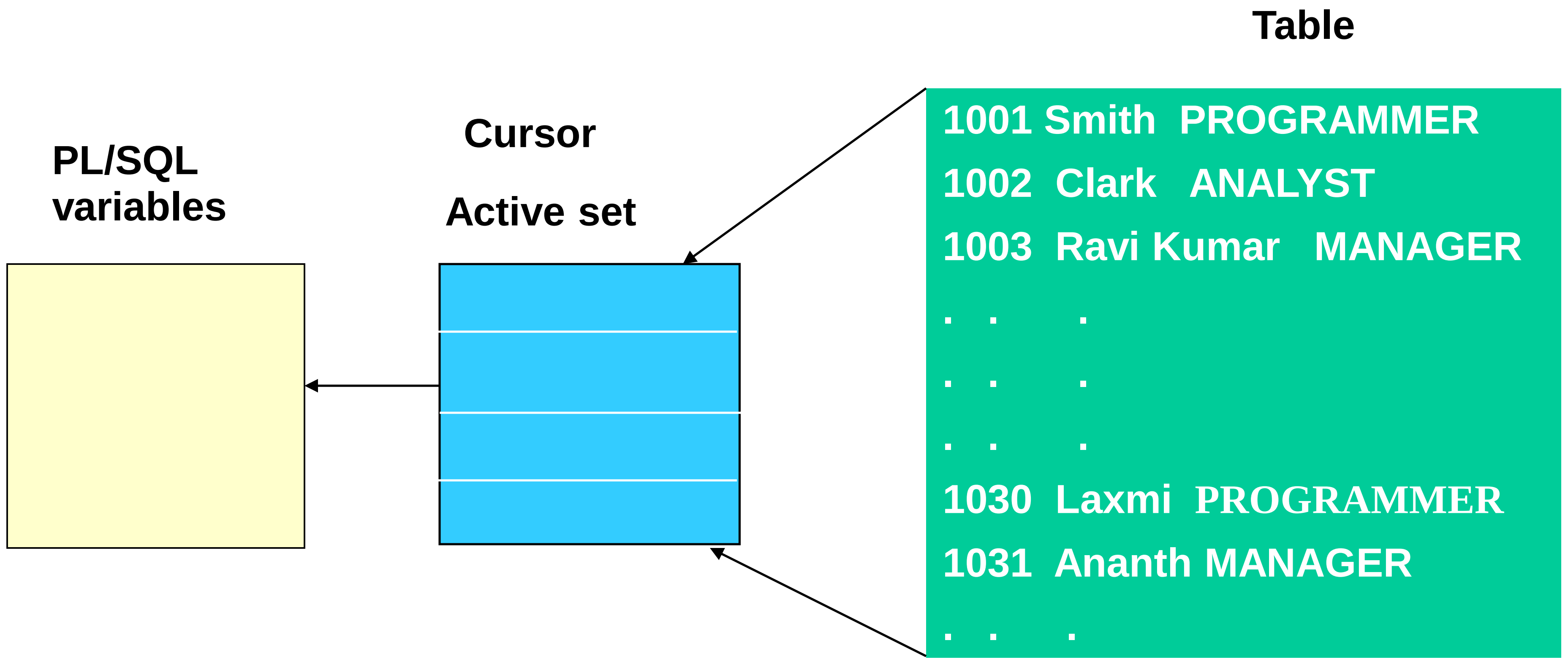
```
   LOOP
           FETCH CURSOR_NAME INTO VARIABLE(S);
           EXIT WHEN CURSOR _NAME %NOT FOUND;
   END LOOP;
```

NOTE: THE STATEMENT AFTER FETCH STATEMENT HAS TO BE EXIT WHEN STATEMENT.

4. CLOSING THE CURSOR:

```
            CLOSE CURSOR_NAME;
```

Explicit Cursors



Explicit Cursors

```
Declare
 cursor c_emp
 is
 select empno,ename,deptno from emp;
 v_empno emp.empno%type;
 v_ename emp.ename%type;
  v_deptno emp.deptno%type;
Begin
 open c_emp;
    Loop
          fetch c_emp into v_empno,v_ename,v_deptno ;
          exit when c_emp%notfound;
          DBMS_OUTPUT.PUT_LINE (v_empno ||', ' || v_ename ||','||v_deptno);
    End Loop;
End;
```

Explicit Cursors

```
Declare
  cursor c_emp
  is
  select * from emp;
  v_emprec  emp%rowtype;
Begin
 open c_emp;
   Loop
         fetch c_emp into v_emprec ;
        exit when c_emp%notfound;
    DBMS_OUTPUT.PUT_LINE (v_emprec.empno ||', ' || v_emprec.ename
                                                    ||','||v_emprec.deptno);
   End Loop;
End;
```

Explicit Cursors

```
Declare
 cursor c_emp
 is
 select empno,ename,deptno from emp;
 v_erec  c_emp%rowtype; -- cursor rowtype
Begin
 open c_emp;
    Loop
       fetch c_emp into v_erec ;
       exit when c_emp%notfound;
       DBMS_OUTPUT.PUT_LINE (v_erec.empno ||', ' ||
           v_erec.ename ||','||v_erec.deptno);
    End Loop;
End;
```

Explicit Cursors

```
DECLARE
 CURSOR C1  IS
 SELECT EMPNO , NVL(SAL,0), NVL(COMM,0) FROM EMP;
 V_SAL   EMP.SAL%TYPE;
 V_COMM  EMP.COMM%TYPE;
 V_EMPNO  EMP.EMPNO%TYPE;
 V_TOTALPAY  V_SAL%TYPE;
BEGIN
 OPEN C1;
 LOOP
           FETCH C1 INTO V_EMPNO,V_SAL,V_COMM;
           EXIT WHEN C1%NOTFOUND;
          V_TOTALPAY := V_SAL + V_COMM;
     DBMS_OUTPUT.PUT_LINE('TOTAL PAY OF EMPNO:'  || V_EMPNO ||  '=' || V_TOTALPAY);
 END LOOP;
 CLOSE C1;
END;
```

PARAMETERIZED CURSORS

CURSOR *cursorname* [(*parameter_name datatype*, ...)]
IS
***select statement*;**

Note: datatypes have to be unconstrained

OPEN *cursorname(parameter value*,.....) ;

```
 DECLARE
  CURSOR C1(P_DNO NUMBER)  -- Formal parameter
    IS
   SELECT * FROM EMP
  WHERE DEPTNO = P_DNO;
  V_TOTAL EMP.SAL%TYPE := 0;
  V_EMPREC EMP%ROWTYPE;
 BEGIN
  OPEN C1(10);  -- Actual Parameter
  LOOP
            FETCH C1 INTO V_EMPREC;
            EXIT WHEN C1%NOTFOUND;
 V_TOTAL := V_TOTAL +  NVL(V_EMPREC.SAL,0) + NVL(V_EMPREC.COMM,0);
  END LOOP;
  CLOSE C1;
DBMS_OUTPUT.PUT_LINE( 'PAY ROLL OF DEPT:' || V_EMPREC.DEPTNO|| '='|| V_TOTAL);
END;
```

WHERE CURRENT OF CLAUSE

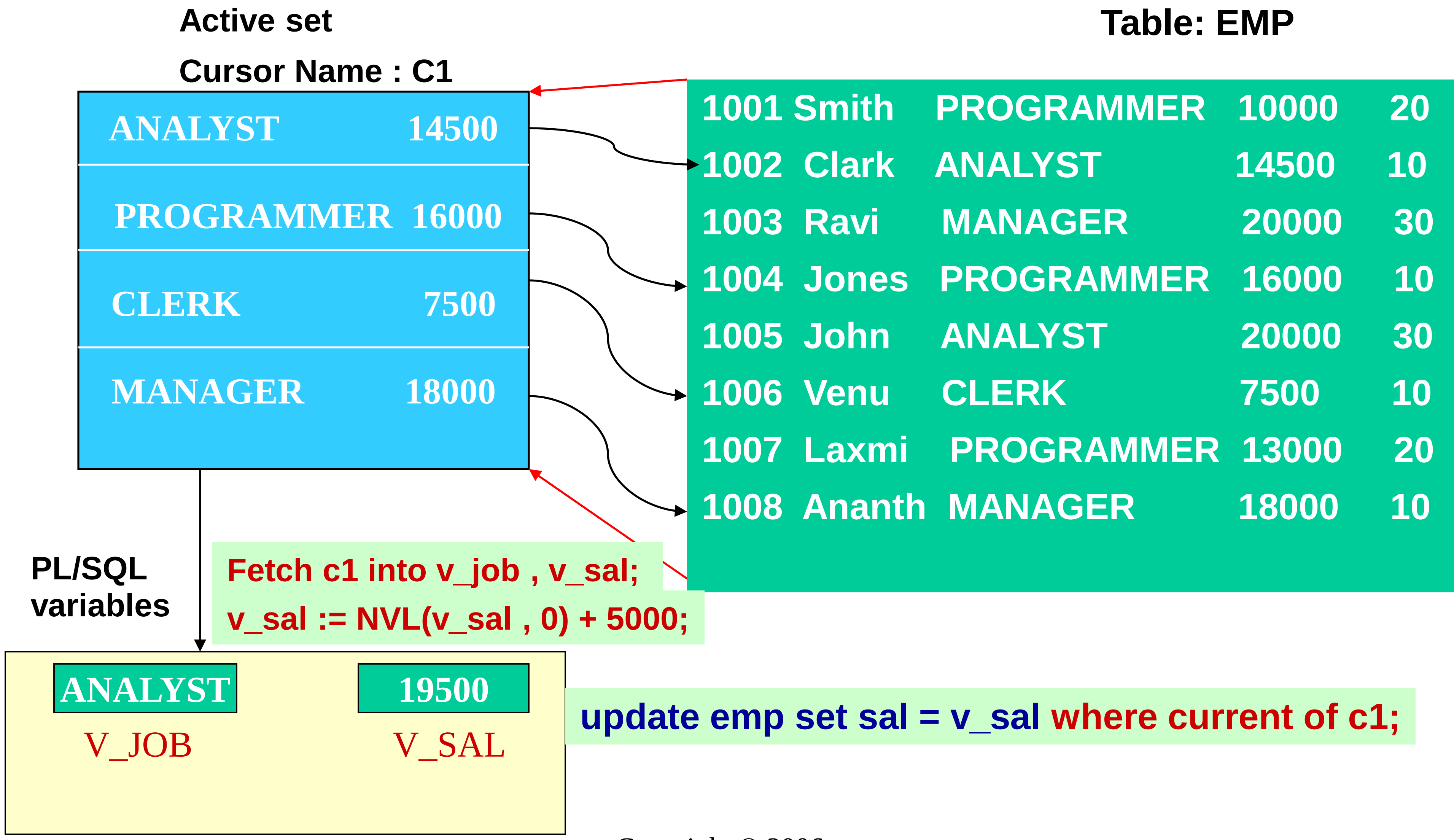
WHEN REFERENCING THE CURRENT ROW FROM EXPLICIT CURSOR, SQL COMMANDS CAN USE
” WHERE CURRENT OF” CLAUSE.

THIS ALLOWS CHANGES TO BE APPLIED ON A SINGLE ROW WHICH IS CURRENTLY BEING ADDRESSED, WITHOUT THE NEED TO EXPLICITLY REFERENCE ROWID.

NOTE:
SPECIFY “FOR UPDATE“ CLAUSE IN THE CURSOR’S QUERY.

Explicit Cursors

```
CURSOR C1 is SELECT JOB, SAL FROM EMP WHERE DEPTNO = 10 FOR UPDATE ;
```



Active set — Cursor Name : C1

ANALYST	14500
PROGRAMMER	16000
CLERK	7500
MANAGER	18000

Table: EMP

1001	Smith	PROGRAMMER	10000	20
1002	Clark	ANALYST	14500	10
1003	Ravi	MANAGER	20000	30
1004	Jones	PROGRAMMER	16000	10
1005	John	ANALYST	20000	30
1006	Venu	CLERK	7500	10
1007	Laxmi	PROGRAMMER	13000	20
1008	Ananth	MANAGER	18000	10

```
Fetch c1 into v_job , v_sal;
v_sal := NVL(v_sal , 0) + 5000;
```

```
update emp set sal = v_sal where current of c1;
```

WHERE CURRENT OF CLAUSE

```
DECLARE
        CURSOR C1   IS   SELECT  job, sal FROM EMP FOR UPDATE ;
         v_sal emp.sal%TYPE;
         v_job emp.job%type;
BEGIN
         OPEN C1;
         LOOP
                 FETCH C1 INTO v_job, v_sal;
                 EXIT WHEN C1%NOTFOUND;
                 IF upper(v_job) = 'MANAGER' THEN
                         v_sal := NVL(v_sal , 0) + 5000;
                 ELSIF upper(v_job) = 'PROGRAMMER' THEN
                         v_sal:= NVL(v_sal , 0) + 3000;
                 ELSE
                         v_sal := NVL(v_sal , 0) + 2000;
                 END  IF;
```

```
          UPDATE  EMP   SET SAL = v_sal   WHERE CURRENT OF C1;
          END LOOP;
          CLOSE C1;
END;
```

CURSOR FOR LOOP

CURSOR FOR LOOP IMPLICITLY DECLARES ITS LOOP COUNTER AS %ROWTYPE,OPENS THE CURSOR,REPEATEDLY FETCHES ROWS FROM THE ACTIVE SET INTO THE VARIABLES AND CLOSES THE CURSOR WHEN ALL ROWS ARE PROCESSED.

SYNTAX:

```
FOR LOOP_ COUNTER IN CURSOR_NAME
 LOOP
         ------------------ ;
         ------------------  ;
END LOOP;
```

CURSOR FOR LOOP

```
DECLARE
               CURSOR C1(p_dno NUMBER)
                IS
                SELECT * FROM EMP
                WHERE DEPTNO= p_dno;

                V_TOTAL NUMBER(10,2) := 0;

 BEGIN
                FOR EREC IN C1(10)
                LOOP

                  V_TOTAL := V_TOTAL + NVL(EREC.SAL,0)+NVL(EREC.COMM,0);

                END LOOP;

                DBMS_OUTPUT.PUT_LINE( 'PAY ROLL = ' || V_TOTAL);

 END;
/
```

CURSOR FOR LOOP

```
DECLARE

            CURSOR C1
            IS
            SELECT empno,ename,job,sal,e.deptno,dname,loc
            FROM EMP e , dept d
            WHERE e.deptno= d.deptno;

 BEGIN

            FOR C_REC IN C1
            LOOP
            DBMS_OUTPUT.PUT_LINE (c_rec.empno||' , '|| c_rec.ename || ', '|| c_rec.dname);
            END LOOP;

 END;
```

CURSOR FOR LOOP USING SUBQUERY

```
DECLARE
 V_TOTAL  EMP.SAL%TYPE := 0;
BEGIN
 FOR EMP_REC IN (SELECT  * FROM EMP WHERE DEPTNO=10)
 LOOP
 V_TOTAL := V_TOTAL+ NVL(EMP_REC.SAL,0)+NVL(EMP_REC.COMM,0);
 END LOOP;
 DBMS_OUTPUT.PUT_LINE( 'PAY ROLL = ' || V_TOTAL);
END;
```

Cursor Variables

Cursor variables are pointers to the Active Set I.e Area in the memory where the cursor data is stored.

Cursor variables enable us to create dynamic cursors.

When we create a cursor variable , we are creating a dynamic cursor because we specify the select statement when we open the cursor.

The same cursor variable can be reopened with a different query.

Cursor Variables

I. Declaration of a cursor variable

There are 2 steps involved:

1. Define refcursor type

```
Type refcursor_type is ref cursor;
```

2. Declare cursor variable

```
 cursor_variable  refcursor_type;
```

Cursor Variables

2. Opening cursor variable

```
open cursor_variable for select statement;
```

3. Closing the cursor variable

```
close cursor_variable ;
```

NOTE: We cannot use cursor for loops with cursor variables

Cursor Variables

```
Declare
   Type GenCurTyp is ref cursor;
   gcv GenCurTyp;
   emp_rec  emp%rowtype;
   dept_rec  dept%rowtype;

Begin
 open gcv  for select * from  emp;
 loop
   fetch gcv into  emp_rec;
   exit when gcv%notfound;
   dbms_output.put_line (emp_rec.empno ||','|| emp_rec.ename);
  end loop;

open gcv  for select * from  dept;
 loop
   fetch gcv into  dept_rec;
   exit when gcv%notfound;
dbms_output.put_line (dept_rec.deptno ||','|| dept_rec.dname);
  end loop;
close gcv;
exception
when others then
  null;
end;
```