```
        EXIT WHEN C1%NOTFOUND;
        SELECT SAL INTO OLDSAL FROM EMP WHERE
        EMPNO=REC.EMPNO;
        IF REC.DEPTNO=10 THEN
            UPDATE EMP SET SAL=SAL+1000 WHERE
            EMPNO=REC.EMPNO;
            SELECT SAL INTO NEWSAL FROM EMP WHERE
            EMPNO=REC.EMPNO;
        END IF;
        IF REC.DEPTNO=20 THEN
        UPDATE EMP SET SAL=SAL+500 WHERE
    EMPNO=REC.EMPNO;
        SELECT SAL INTO NEWSAL FROM EMP WHERE
EMPNO=REC.EMPNO;
    END IF;
    IF REC.DEPTNO=30 THEN
        UPDATE EMP SET SAL=SAL+800 WHERE
    EMPNO=REC.EMPNO;
        SELECT SAL INTO NEWSAL FROM EMP WHERE
    EMPNO=REC.EMPNO;
    END IF;
    INSERT INTO TEMP(EMPNO,OLD,NEW)
    VALUES(REC.EMPNO,OLDSAL,NEWSAL);
END LOOP;
CLOSE C1;
END;
```

Example 21.7: *Consider a PL/SQL code to calculate the total and percentage of marks of the students in four subjects from table result, which has the following structure:*

(rno,s1,s2,s3,s4,total,percentage).

The rollno and marks in each subject are stored in the database, PL/SQL code calculate the total and percentage of each student and update the database.

Solution:

```
DECLARE
    T NUMBER;
    PER NUMBER;
```

```
    CURSOR C1 IS SELECT * FROM RESULT;
    REC C1%ROWTYPE;
BEGIN
    OPEN C1;
    LOOP
        FETCH C1 INTO REC;
        EXIT WHEN C1%NOTFOUND;
        T:=REC.S1+REC.S2+REC.S3+REC.S4;
        PER:=T/4;
        UPDATE RESULT SET TOTAL=T, PERCENTAGE=PER WHERE
    RNO=REC.RNO;
    END LOOP;
    CLOSE C1;
END;
```

21.4 CURSOR FOR LOOP

The Cursor FOR Loop implicitly declares its loop index as a record of type %ROWTYPE, opens a cursor, repeatedly fetches rows of the values from the active set into fields in the record, then closes the cursor when all rows have been processed or when the EXIT command is encountered.

The syntax for the FOR Loop is:

```
FOR  <variable name> IN <cursor_name> LOOP
    <statements>;
END LOOP;
```

It is a machine defined loop exit i.e. when all the values in the FOR construct are exhausted looping stops.

A cursor for loop automatically does the following:

- Implicitly declares its loop index or variable_name as a %rowtype record.
- Opens a cursor.
- Fetches a row from the cursor for each loop iteration.
- Closes the cursor when all rows have been processed.

Example 21.8: ***Consider a PL/SQL code to display the empno, ename, job of employees of department number 10 with CURSOR FOR Loop statement.***

```
        EXIT WHEN C1%NOTFOUND;
        SELECT SAL INTO OLDSAL FROM EMP WHERE
        EMPNO=REC.EMPNO;
        IF REC.DEPTNO=10 THEN
            UPDATE EMP SET SAL=SAL+1000 WHERE
            EMPNO=REC.EMPNO;
            SELECT SAL INTO NEWSAL FROM EMP WHERE
            EMPNO=REC.EMPNO;
        END IF;
        IF REC.DEPTNO=20 THEN
        UPDATE EMP SET SAL=SAL+500 WHERE
    EMPNO=REC.EMPNO;
        SELECT SAL INTO NEWSAL FROM EMP WHERE
EMPNO=REC.EMPNO;
    END IF;
    IF REC.DEPTNO=30 THEN
        UPDATE EMP SET SAL=SAL+800 WHERE
    EMPNO=REC.EMPNO;
        SELECT SAL INTO NEWSAL FROM EMP WHERE
    EMPNO=REC.EMPNO;
    END IF;
    INSERT INTO TEMP(EMPNO,OLD,NEW)
    VALUES(REC.EMPNO,OLDSAL,NEWSAL);
END LOOP;
CLOSE C1;
END;
```

Example 21.7: *Consider a PL/SQL code to calculate the total and percentage of marks of the students in four subjects from table result, which has the following structure:*

(rno,s1,s2,s3,s4,total,percentage).

The rollno and marks in each subject are stored in the database, PL/SQL code calculate the total and percentage of each student and update the database.

Solution:

```
DECLARE
    T NUMBER;
    PER NUMBER;
```

Solution:

```
DECLARE
    CURSOR C1 IS SELECT EMPNO, ENAME, JOB FROM EMP WHERE
    DEPTNO=10;
BEGIN
    FOR REC IN C1 LOOP
        DBMS_OUTPUT.PUT_LINE('EMPNO '||REC.EMPNO);
        DBMS_OUTPUT.PUT_LINE('ENAME '||REC.ENAME);
        DBMS_OUTPUT.PUT_LINE('JOB '||REC.JOB);
    END LOOP;
END;
```

Example21.9: *Consider a PL/SQL code to display the employee number and name of top 5 highest paid employees with CURSOR FOR LOOP statement.*

Solution:

```
DECLARE
    CURSOR TEMP1 IS SELECT ENAME, SAL FROM EMP ORDER BY
 SAL DESC;
BEGIN
    FOR REC IN TEMP1 LOOP
        EXIT WHEN TEMP1%ROWCOUNT>5;
        DBMS_OUTPUT.PUT_LINE(REC.ENAME || REC.SAL);
    END LOOP;
END;
```

21.5 CURSORS WITH PARAMETERS

Commercial applications require that the query, which, defines the cursor, be generic and the data that is retrieved from the table be allowed to change according to need. Oracle recognizes this and permits the creation of a parameterized cursor prior opening. Parameters allow values to be passed to a cursor when it is opened, and used within a query when it executes. Using parameters explicit cursor may be opened more than once in a block, returning a different working set each time. The parameters can be either a constant or a variable.

Syntax to declare parameterized cursor:

```
CURSOR cursor_name (variable_name datatype) IS
    <SELECT statement...>
```

Syntax to open a Parameterized cursor and passing values to the cursor:

```
OPEN cursor_name (value/variable/expression);
```

Note: The scope of cursor parameters is local to that cursor, which means that they can be referenced only within the query declared in the cursor declaration. Each parameter in the declaration must have a corresponding value in the open statement.

Example21.10: *Consider a PL/SQL code to calculate the total salary of first n records of emp table. The value of n is passed to cursor as parameter.*

Solution:

```
DECLARE
        E NUMBER(5);
        S NUMBER(5):=0;
        CURSOR C1(N NUMBER) IS SELECT SAL FROM EMP  WHERE
 ROWNUM<=N;
BEGIN
        OPEN C1(&N);
        LOOP
              FETCH C1 INTO E;
              EXIT WHEN C1%NOTFOUND;
              S:=S+E;
        END LOOP;
        DBMS_OUTPUT.PUT_LINE(S);
        CLOSE C1;
END;
```

With Cursor FOR Loop

```
DECLARE
        S NUMBER(5):=0;
        CURSOR C1(N NUMBER) IS SELECT SAL FROM EMP WHERE
        ROWNUM<=N;
BEGIN
        FOR REC IN C1(&N) LOOP
              S:=S+REC.SAL;
        END LOOP;
        DBMS_OUTPUT.PUT_LINE(S);
END;
```