College Of Engineering Trivandrum

# Application Software Development Lab

Abhishek Manoharan
S5 CSE Roll No:2

TVE17CS002

Department of Computer Science

August 19, 2019

# Contents



# List of Figures

**Cycle 2**

**Exp No 9**

CURSOR

# 1 Aim

To study the use and implementation of cursors in PL/SQL.

# 2 Description

A PL/pgSQL cursor allows us to encapsulate a query and process each individual row at a time. We use cursors when we want to divide a large result set into parts and process each part individually. If we process it at once, we may have a memory overflow error.

# 3 Questions

## 3.1 Grade calculation

Create table student (id, name, m1, m2, m3, grade).Insert 5 tuples into it. Find the total, calculate grade and update the grade in the table.

### 3.1.1 Table Creation

**create table stdnt(id int,sname varchar(15),m1 int,m2 int,m3 int,gr char(1));**

```
insert into stdnt (id,sname,m1,m2,m3) values(88,'anu',39,67,92);
insert into stdnt (id,sname,m1,m2,m3) values(10,'jan',58,61,29);
insert into stdnt (id,sname,m1,m2,m3) values(30,'karuna',87,79,77);
insert into stdnt (id,sname,m1,m2,m3) values(29,'jossy',39,80,45);
```

```
select * from stdnt;
```

```
asdlab=# select * from stdnt;
 id | sname  | m1 | m2 | m3 | gr
----+--------+----+----+----+----
 10 | jan    | 58 | 61 | 29 |
 30 | karuna | 87 | 79 | 77 |
 88 | anu    | 39 | 67 | 92 |
 29 | jossy  | 39 | 80 | 45 |
(4 rows)

asdlab=#
```

Figure 1: stdnt Table

#### 3.1.2 Code

```
CREATE OR REPLACE FUNCTION get_grade()
   RETURNS void AS $$
DECLARE
    total INT ;
    grade char(1);
    rec_film   RECORD;
    cur_films CURSOR
       FOR SELECT * FROM stdnt;
BEGIN
   OPEN cur_films;
   LOOP
      FETCH cur_films INTO rec_film;
      EXIT WHEN NOT FOUND;
      total = rec_film.m1+rec_film.m2+rec_film.m3;
      IF total>240 THEN
          grade='A';
      ELSIF total>180 THEN
        grade='B';
      ELSIF total>120 THEN
        grade='C';
     ELSIF total>60 THEN
        grade='D';
    ELSE
            grade='F';
      END IF;
      update stdnt set gr=grade where m1=rec_film.m1;
   END LOOP;
   CLOSE cur_films;
END; $$
LANGUAGE plpgsql;
```

### 3.1.3 Output

```
select get_grade();
select * from stdnt;
```

```
asdlab=# CREATE OR REPLACE FUNCTION get_grade()
   RETURNS void AS $$
DECLARE
    total INT ;
    grade char(1);
    rec_film   RECORD;
    cur_films CURSOR
       FOR SELECT * FROM stdnt;
BEGIN
   OPEN cur_films;
   LOOP
      FETCH cur_films INTO rec_film;
      EXIT WHEN NOT FOUND;
      total = rec_film.m1+rec_film.m2+rec_film.m3;
      IF total>240 THEN
           grade='A';
      ELSIF total>180 THEN
  grade='B';
      ELSIF total>120 THEN
  grade='C';
     ELSIF total>60 THEN
  grade='D';
    ELSE
 grade='F';
      END IF;
      update stdnt set gr=grade where m1=rec_film.m1;
   END LOOP;
   CLOSE cur_films;

END; $$

LANGUAGE plpgsql;
CREATE FUNCTION
asdlab=# select from get_grade();
--
(1 row)

asdlab=# select * from stdnt;
 id | sname  | m1 | m2 | m3 | gr
----+--------+----+----+----+----
 10 | jan    | 58 | 61 | 29 | C
 30 | karuna | 87 | 79 | 77 | A
 88 | anu    | 39 | 67 | 92 | C
 29 | jossy  | 39 | 80 | 45 | C
(4 rows)
```

Figure 2: Assigned Grades

## 3.2 Interest Calculation

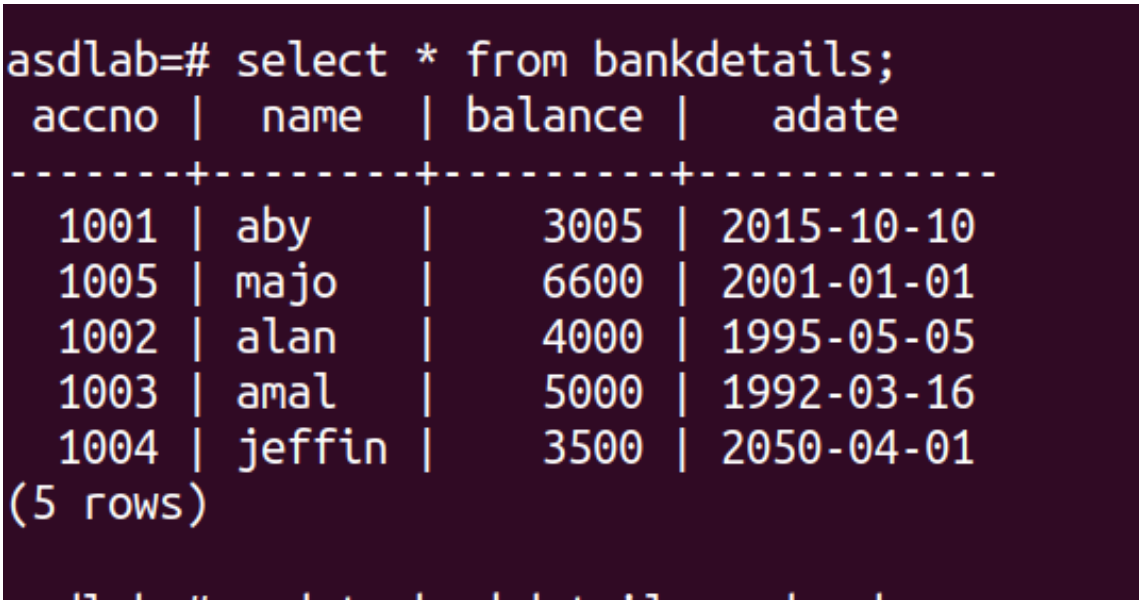
Create bank_details (accno, name, balance, adate).Calculate the interest of the amount and insert into a new table with fields (accno, interest). Interest= 0.08*balance.

### 3.2.1 Table Creation

**create table bankdetails(accno int,name varchar(15),balance int,adate date);**
**create table banknew(accno int,interest int);**

```
insert into bankdetails values(1001,'aby',3005,'10-oct-15');
insert into bankdetails values(1002,'alan',4000,'05-may-95');
insert into bankdetails values(1003,'amal',5000,'16-mar-92');
insert into bankdetails values(1004,'jeffin',3500,'01-apr-50');
insert into bankdetails values(1005,'majo',6600,'01-jan-01');
```

```
select * from bankdetails;
```



```
asdlab=# select * from bankdetails;
 accno |  name  | balance |   adate
-------+--------+---------+------------
  1001 | aby    |    3005 | 2015-10-10
  1005 | majo   |    6600 | 2001-01-01
  1002 | alan   |    4000 | 1995-05-05
  1003 | amal   |    5000 | 1992-03-16
  1004 | jeffin |    3500 | 2050-04-01
(5 rows)
```

Figure 3: bankdetails Table

### 3.2.2 Code

```
CREATE OR REPLACE FUNCTION get_interest()
   RETURNS void AS $$
DECLARE
    interest INT ;
    account   RECORD;
    movacc CURSOR
       FOR SELECT * FROM bankdetails;
BEGIN
   OPEN movacc;
   LOOP
      FETCH movacc INTO account;
      EXIT WHEN NOT FOUND;

      interest=0.08*account.balance;
      INSERT INTO banknew VALUES (account.accno,interest);
   END LOOP;
   CLOSE movacc;

END; $$

LANGUAGE plpgsql;
```

### 3.2.3 Output

```
select get_interest();
select * from banknew;
```

```
asdlab=# CREATE OR REPLACE FUNCTION get_interest()
asdlab-#    RETURNS void AS $$
asdlab$# DECLARE
asdlab$#     interest INT ;
asdlab$#     account   RECORD;
asdlab$#     movacc CURSOR
asdlab$#        FOR SELECT * FROM bankdetails;
asdlab$# BEGIN
asdlab$#    OPEN movacc;
asdlab$#    LOOP
asdlab$#       FETCH movacc INTO account;
asdlab$#       EXIT WHEN NOT FOUND;
asdlab$#
asdlab$#       interest=0.08*account.balance;
asdlab$#       INSERT INTO banknew VALUES (account.accno,interest);
asdlab$#    END LOOP;
asdlab$#    CLOSE movacc;
asdlab$#
asdlab$# END; $$
asdlab-#
asdlab-# LANGUAGE plpgsql;
CREATE FUNCTION
asdlab=# select from get_interest();
--
(1 row)

asdlab=# select * from banknew;
 accno | interest
-------+----------
  1001 |      240
  1005 |      528
  1002 |      320
  1003 |      400
  1004 |      280
(5 rows)

asdlab=#
```

Figure 4: banknew table

### 3.3 Finding Experienced People

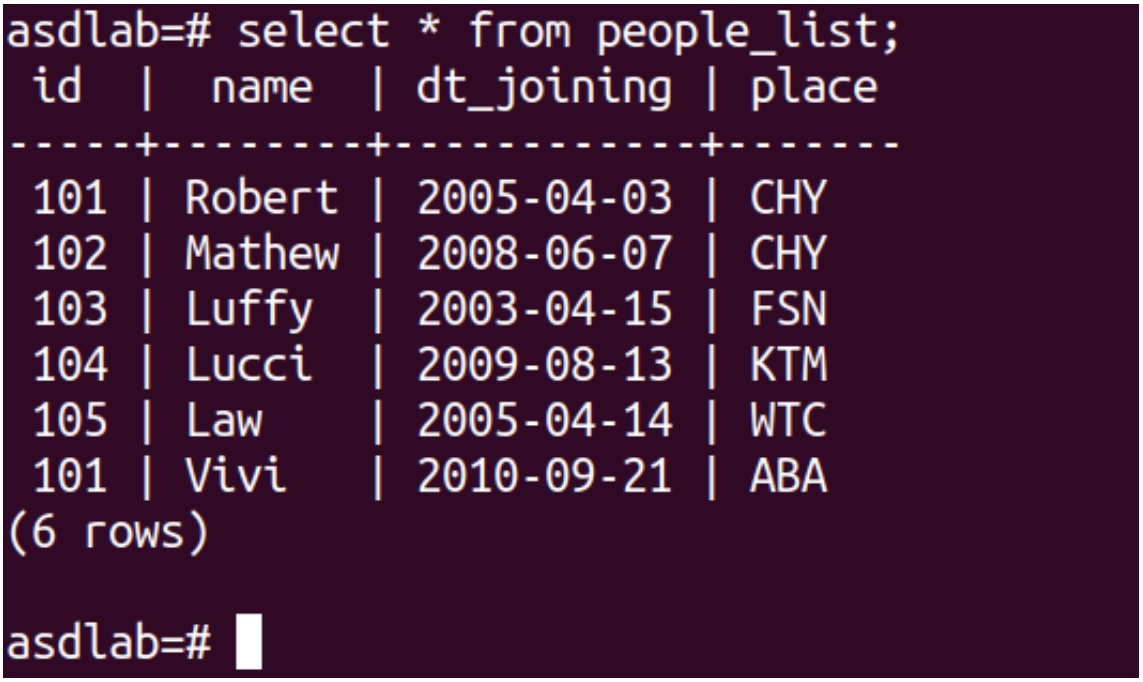
**Create table people_list (id, name, dt_joining, place).If person's experience is above 10 years, put the tuple in table exp_list (id, name, experience).)**

#### 3.3.1 Table creation

**create table people_list(id INT, name varchar(20),dt_joining DATE,place varchar(20));**
**create table exp_list(id INT, name varchar(20),exp INT);**

```
insert into people_list values(101,'Robert','03-APR-2005','CHY');
insert into people_list values(102,'Mathew','07-JUN-2008','CHY');
insert into people_list values(103,'Luffy','15-APR-2003','FSN');
insert into people_list values(104,'Lucci','13-AUG-2009','KTM');
insert into people_list values(105,'Law','14-APR-2005','WTC');
insert into people_list values(101,'Vivi','21-SEP-2010','ABA');
```

```
select * from people_list;
```



```
asdlab=# select * from people_list;
 id  |  name  | dt_joining | place
-----+--------+------------+-------
 101 | Robert | 2005-04-03 | CHY
 102 | Mathew | 2008-06-07 | CHY
 103 | Luffy  | 2003-04-15 | FSN
 104 | Lucci  | 2009-08-13 | KTM
 105 | Law    | 2005-04-14 | WTC
 101 | Vivi   | 2010-09-21 | ABA
(6 rows)

asdlab=#
```

Figure 5: people_list Table

#### 3.3.2 Code

```
CREATE OR REPLACE FUNCTION set_exp()
   RETURNS void AS $$
DECLARE
    exp INT;
    proff   RECORD;
    today DATE;
    movproff CURSOR
       FOR SELECT * FROM people_list;
BEGIN
   OPEN movproff;
   SELECT current_date INTO today;
   LOOP
      FETCH movproff INTO proff;
      EXIT WHEN NOT FOUND;

      SELECT DATE_PART('year', today::date) - DATE_PART('year', proff.dt_joining::date)
      IF exp>10 THEN
       INSERT INTO exp_list VALUES (proff.id,proff.name,exp);
      END IF;
   END LOOP;
   CLOSE movproff;

END; $$

LANGUAGE plpgsql;
```

### 3.3.3 Output

```
select from set_exp;
select * from exp_list;
```

```
asdlab=# CREATE OR REPLACE FUNCTION put_exp()
   RETURNS void AS $$
DECLARE
    exp INT;
    proff   RECORD;
    today DATE;
    movproff CURSOR
       FOR SELECT * FROM people_list;
BEGIN
   OPEN movproff;
   SELECT current_date INTO today;
   LOOP
      FETCH movproff INTO proff;
      EXIT WHEN NOT FOUND;

      SELECT DATE_PART('year', today::date) - DATE_PART('year', proff.dt_joining::date) INTO exp;
      IF exp>10 THEN
       INSERT INTO exp_list VALUES (proff.id,proff.name,exp);
      END IF;
   END LOOP;
   CLOSE movproff;

END; $$

LANGUAGE plpgsql;
CREATE FUNCTION
asdlab=# select put_exp();
 put_exp
---------

(1 row)

asdlab=# select * from exp_list;
 id  |  name  | exp
-----+--------+-----
 101 | Robert |  14
 102 | Mathew |  11
 103 | Luffy  |  16
 105 | Law    |  14
(4 rows)
```

Figure 6: more than 10 experienced

### 3.4 Salary Increment

**Create table employee_list(id,name,monthly salary).**
**If: annual salary ¡ 60000, increment monthly salary by 25%**
**between 60000 and 200000, increment by 20%**
**between 200000 and 500000, increment by 15%**
**annual salary ¿ 500000, increment monthly salary by 10%.**

#### 3.4.1 Table creation

**create table emp_list(id INT,Name varchar(20),M_sal INT);**

```
  insert into emp_list values(101,'Mathew',55000);
insert into emp_list values(102,'Jose',80000);
insert into emp_list values(103,'John',250000);
insert into emp_list values(104,'Ann',600000);

  select * from emp_list;
```

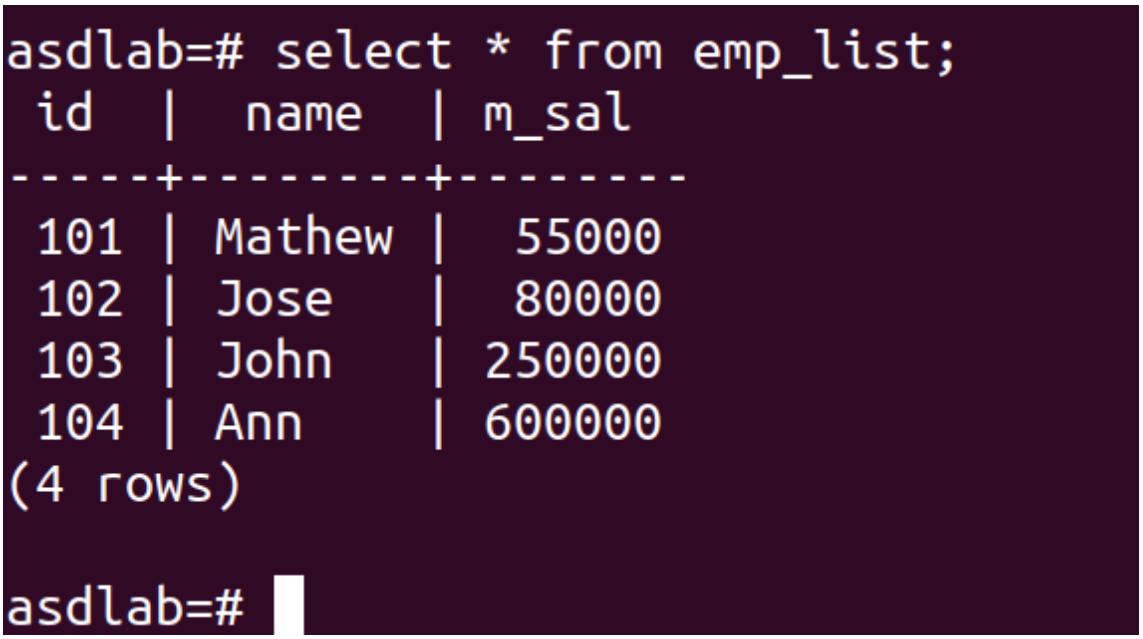



Figure 7: emp_list Table

### 3.4.2 Code

```
CREATE OR REPLACE FUNCTION sal_incre()
   RETURNS void AS $$
DECLARE
    yearsal INT;
    monsal INT;
    sal   RECORD;
    movsal CURSOR
       FOR SELECT * FROM emp_list;
BEGIN
   OPEN movsal;
   LOOP
      FETCH movsal INTO sal;
      EXIT WHEN NOT FOUND;
      yearsal=sal.m_sal*12;
      monsal=sal.m_sal;

        IF yearsal>500000 THEN
           UPDATE emp_list SET m_sal=monsal*1.1 WHERE m_sal=monsal;
        ELSIF yearsal>200000 THEN
        UPDATE emp_list SET m_sal=monsal*1.15 WHERE m_sal=monsal;
        ELSIF yearsal>60000 THEN
         UPDATE emp_list SET m_sal=monsal*1.2WHERE m_sal=monsal;
        ELSE
         UPDATE emp_list SET m_sal=monsal*1.25 WHERE m_sal=monsal;
     END IF;
   END LOOP;
   CLOSE movsal;

END; $$

LANGUAGE plpgsql;
```

### 3.4.3 Output

```
select from sal_incre();
select * from emp_list;
```

```
asdlab=# CREATE OR REPLACE FUNCTION sal_incre()
asdlab-#    RETURNS void AS $$
asdlab$# DECLARE
asdlab$#     yearsal INT;
asdlab$#     monsal INT;
asdlab$#     sal   RECORD;
asdlab$#     movsal CURSOR
asdlab$#        FOR SELECT * FROM emp_list;
asdlab$# BEGIN
asdlab$#    OPEN movsal;
asdlab$#    LOOP
asdlab$#       FETCH movsal INTO sal;
asdlab$#       EXIT WHEN NOT FOUND;
asdlab$#       yearsal=sal.m_sal*12;
asdlab$#       monsal=sal.m_sal;
asdlab$#
asdlab$#       IF yearsal>500000 THEN
asdlab$#        UPDATE emp_list SET m_sal=monsal*1.1 WHERE m_sal=monsal;
asdlab$#      ELSIF yearsal>200000 THEN
asdlab$#  UPDATE emp_list SET m_sal=monsal*1.15 WHERE m_sal=monsal;
asdlab$#     ELSIF yearsal>60000 THEN
asdlab$#  UPDATE emp_list SET m_sal=monsal*1.2WHERE m_sal=monsal;
asdlab$#    ELSE
asdlab$#  UPDATE emp_list SET m_sal=monsal*1.25 WHERE m_sal=monsal;
asdlab$#       END IF;
asdlab$#    END LOOP;
asdlab$#    CLOSE movsal;
asdlab$#
asdlab$# END; $$
asdlab-#
asdlab-# LANGUAGE plpgsql;
CREATE FUNCTION
asdlab=# select from sal_incre();
--
(1 row)
```

Figure 8: salary increment function

```
asdlab=# select * from emp_list;
 id  |  name   | m_sal
-----+---------+--------
 101 | Mathew  |  60500
 102 | Jose    |  88000
 103 | John    | 275000
 104 | Ann     | 660000
(4 rows)

asdlab=#
```

Figure 9: emp_list table updated

## 4 Result

The PL/SQL program was executed successfully and the output was obtained.