College Of Engineering Trivandrum

Application Software Development Lab

Abhishek Manoharan
S5 CSE Roll No:2

TVE17CS002

Department of Computer Science

August 19, 2019

Cycle 2

Exp No 8

PL/PGSQL AND SEQUENCE

1 Aim

To study the basic pl/pgsql and sequence queries.

2 Description

PL/pgSQL is a loadable procedural language for the PostgreSQL database system. The design goals of PL/pgSQL were to create a loadable procedural language that

- can be used to create functions and trigger procedures,
- adds control structures to the SQL language,
- can perform complex computations,
- inherits all user-defined types, functions, and operators,
- can be defined to be trusted by the server, is easy to use.

Functions created with PL/pgSQL can be used anywhere that built-in functions could be used. For example, it is possible to create complex conditional computation functions and later use them to define operators or use them in index expressions.

In PostgreSQL 9.0 and later, PL/pgSQL is installed by default. However it is still a loadable module, so especially security-conscious administrators could choose to remove it.

3 Questions

Write PL/SQL programs for the following:

3.1 To print the first 'n' prime numbers..

3.1.1 Code

```
CREATE or REPLACE FUNCTION prime(prime INT) RETURNS VOID as
$$
DECLARE
flag INT;
count INT =0;
i INT;
start INT=2;
rem INT;
BEGIN
WHILE (count<prime) LOOP
flag =0;
FOR i IN 2..(start/2) LOOP
rem=start%i;
IF  rem = 0 THEN
flag = 1;
END IF;
END LOOP;
IF flag = 0 THEN
RAISE NOTICE   ' % ' , start;
count=count+1;
END IF;
start=start +1;
END LOOP;END;
$$ LANGUAGE plpgsql;
```

3.1.2 Output

```
select from prime(5);
```

```
postgres=# CREATE or REPLACE FUNCTION prime(prime INT) RETURNS VOID as
$$
DECLARE
 flag INT;
 count INT =0;
 i INT;
 start INT=2;
 rem INT;
BEGIN
 WHILE (count<prime) LOOP
  flag =0;
  FOR i IN 2..(start/2) LOOP
   rem=start%i;
   IF  rem = 0 THEN
    flag = 1;
   END IF;
  END LOOP;
  IF flag = 0 THEN
   RAISE NOTICE   ' % ' , start;
   count=count+1;
  END IF;
  start=start +1;
 END LOOP;
END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
postgres=# select from prime(5);
NOTICE:   2
NOTICE:   3
NOTICE:   5
NOTICE:   7
NOTICE:   11
--
(1 row)
```

Figure 1: N prime numbers

3.2 Display the Fibonacci series upto ‘n’ terms

3.2.1 Code

```
CREATE or REPLACE FUNCTION fib(fib INT) RETURNS VOID as
$$
DECLARE
value INT=1;
new INT=1;
temp INT;
count INT;
BEGIN
RAISE NOTICE ' %', value;
RAISE NOTICE ' %', new;
FOR count in 3..fib LOOP
temp=new;
new=new+value;
value=temp;
RAISE NOTICE ' %', new;
END LOOP;
END;
$$ LANGUAGE plpgsql;
```

3.2.2 Output

```
select * from fib(10);
```

```
postgres=# CREATE or REPLACE FUNCTION fib(fib INT) RETURNS VOID as
postgres-# $$
postgres$# DECLARE
postgres$# value INT=1;
postgres$# new INT=1;
postgres$# temp INT;
postgres$# count INT =1;
postgres$# BEGIN
postgres$# RAISE NOTICE ' %', value;
postgres$# RAISE NOTICE ' %', new;
postgres$# FOR count in 3..fib LOOP
postgres$# temp=new;
postgres$# new=new+value;
postgres$# value=temp;
postgres$# RAISE NOTICE ' %', new;
postgres$# END LOOP;
postgres$# END;
postgres$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
postgres=# select from fib(10);
NOTICE:   1
NOTICE:   1
NOTICE:   2
NOTICE:   3
NOTICE:   5
NOTICE:   8
NOTICE:   13
NOTICE:   21
NOTICE:   34
NOTICE:   55
--
(1 row)
```

Figure 2: N terms in Fibonacci sequence

3.3 Assigning Grade

Create a table named student_grade with the given attributes:
roll, name ,mark1,mark2,mark3, grade. Read the roll, name and marks from the user.
Calculate the grade of the student and insert a tuple into the table using PL/SQL.
(Grade= 'PASS' if AVG ¿40, Grade='FAIL' otherwise)

create table student_grade(roll int primary key,name varchar(10),mark1 int,mark2 int, mark3 int ,grade varchar(4));

```
insert into student_grade values(1,'anu',50,45,48),(2,'manu',50,50,50),
(3,'ramu',35,40,40);

select * from student_grade;
```

```
asdlab=# select * from student_grade;
 roll | name | mark1 | mark2 | mark3 | grade
------+------+-------+-------+-------+-------
    1 | anu  |    50 |    45 |    48 |
    2 | manu |    50 |    50 |    50 |
    3 | ramu |    35 |    40 |    40 |
(3 rows)
```

Figure 3: student_grade Table

3.3.1 Code

```
CREATE or REPLACE FUNCTION set_student_grade() RETURNS VOID as
$$
BEGIN
update student_grade
set grade='pass'
where (mark1+mark2+mark3)/3  >40;
update student_grade
set grade='fail'
where (mark1+mark2+mark3)/3 <=40;
END;
$$ LANGUAGE plpgsql;
```

3.3.2 Output

```
select from set_student_grade;
select * from student_grade;
```

```
asdlab=# CREATE or REPLACE FUNCTION set_student_grade() RETURNS VOID as
asdlab-# $$
asdlab$# BEGIN
asdlab$# update student_grade
asdlab$# set grade='pass'
asdlab$# where (mark1+mark2+mark3)/3  >40;
asdlab$# update student_grade
asdlab$# set grade='fail'
asdlab$# where (mark1+mark2+mark3)/3 <=40;
asdlab$# END;
asdlab$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
asdlab=# select set_student_grade();
 set_student_grade
-------------------

(1 row)

asdlab=# select * from student_grade;
 roll | name | mark1 | mark2 | mark3 | grade
------+------+-------+-------+-------+-------
    1 | anu  |    50 |    45 |    48 | pass
    2 | manu |    50 |    50 |    50 | pass
    3 | ramu |    35 |    40 |    40 | fail
(3 rows)

asdlab=#
```

Figure 4: After setting Grade

3.4 Circle and Area

Create table circle_area (rad,area). For radius 5,10,15,20 25., find the area and insert the corresponding values into the table by using loop structure in PL/SQL.

3.4.1 Code

```
CREATE or REPLACE FUNCTION create_circle() RETURNS VOID as
$$
DECLARE
data1 INT =5;
data2 REAL;
count INT =5;
i INT ;
BEGIN
CREATE TABLE circle(rad INT ,area REAL);
FOR i IN 1..count LOOP
data2=3.1416*data1*data1;
INSERT INTO circle VALUES(data1,data2);
data1=data1+5;
END LOOP;
END;
$$ LANGUAGE plpgsql;
```

3.4.2 Output

```
select from create_circle();
select * from circle;
```

```
asdlab=# CREATE or REPLACE FUNCTION create_circle() RETURNS VOID as
$$
DECLARE
data1 INT =5;
data2 REAL;
count INT =5;
i INT ;
BEGIN
 CREATE TABLE circle(rad INT ,area REAL);
 FOR i IN 1..count LOOP
  data2=3.1416*data1*data1;
  INSERT INTO circle VALUES(data1,data2);
  data1=data1+5;
 END LOOP;
END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
asdlab=# select from create_circle();
--
(1 row)

asdlab=# select  * from circle;
 rad |  area
-----+---------
   5 |   78.54
  10 |  314.16
  15 |  706.86
  20 | 1256.64
  25 |  1963.5
(5 rows)

asdlab=#
```

Figure 5: Circle

3.5 Insertion using Array

Use an array to store the names, marks of 10 students in a class. Using Loop structures in PL/SQL insert the ten tuples to a table named stud

```
create table stud(name varchar(10),mark int);
```

3.5.1 Code

```
CREATE or REPLACE FUNCTION insert_stud() RETURNS VOID as
$$
DECLARE
data1 INT []= '{ 25,76,43,45,67,57,97,56,89,8 }';
 data2  VARCHAR(20)[]='{ ARUN,AMAL,PETER,JOSE,ANNIE,MARY,JOSEPH,MARK,MIDHUN,KEVIN}';
i INT ;
BEGIN
FOR i IN 1..10 LOOP
INSERT INTO stud VALUES(data2[i],data1[i]);
END LOOP;
END;
$$ LANGUAGE plpgsql;
```

3.5.2 Output

```
select from insert_stud();
select * from stud;
```

```
asdlab=# CREATE or REPLACE FUNCTION insert_stud() RETURNS VOID as
asdlab-# $$
asdlab$# DECLARE
asdlab$# data1 INT []= '{ 25,76,43,45,67,57,97,56,89,8 }';
asdlab$#  data2  VARCHAR(20)[]='{ ARUN,AMAL,PETER,JOSE,ANNIE,MARY,JOSEPH,MARK,MIDHUN,KEVIN}';
asdlab$# i INT ;
asdlab$# BEGIN
asdlab$#  FOR i IN 1..10 LOOP
asdlab$#   INSERT INTO stud VALUES(data2[i],data1[i]);
asdlab$#  END LOOP;
asdlab$# END;
asdlab$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
asdlab=# select insert_stud();
 insert_stud
-------------

(1 row)

asdlab=# select * from stud;
  name  | mark
--------+------
 ARUN   |   25
 AMAL   |   76
 PETER  |   43
 JOSE   |   45
 ANNIE  |   67
 MARY   |   57
 JOSEPH |   97
 MARK   |   56
 MIDHUN |   89
 KEVIN  |    8
(10 rows)

asdlab=#
```

Figure 6: Array insertion

3.6 Insertion using Array

a sequence using PL/SQL. Use this sequence to generate the primary key values for a table named class_cse with attributes roll,name and phone. Insert some tuples using PL/SQL programming.

```
Create table class_cse (roll int primary key,name varchar(10),phone varchar(15));
```

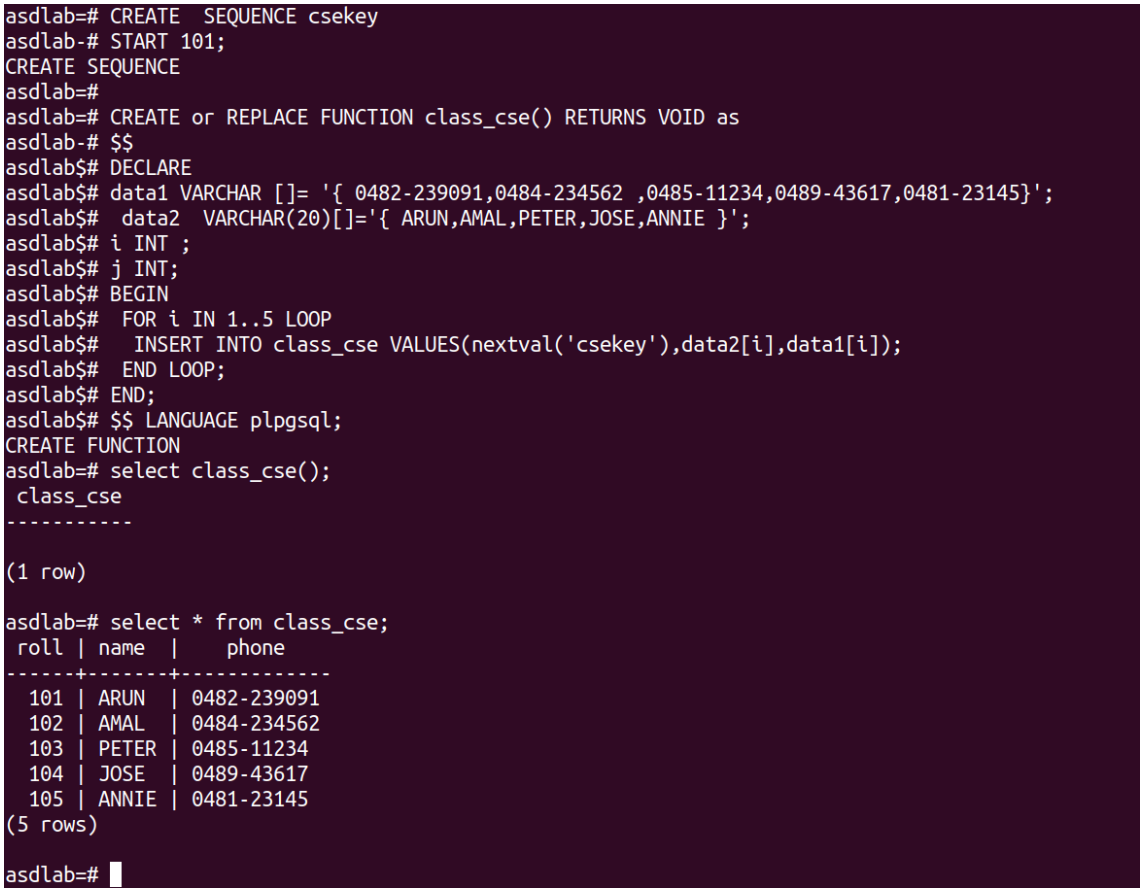
3.6.1 Code

```
CREATE  SEQUENCE csekey
START 101;

CREATE or REPLACE FUNCTION class_cse() RETURNS VOID as
$$
DECLARE
data1 VARCHAR []= '{ 0482-239091,0484-234562 ,0485-11234,0489-43617,0481-23145}';
 data2  VARCHAR(20)[]='{ ARUN,AMAL,PETER,JOSE,ANNIE }';
i INT ;
j INT;
BEGIN
FOR i IN 1..5 LOOP
INSERT INTO class_cse VALUES(nextval('csekey'),data2[i],data1[i]);
END LOOP;
END;
$$ LANGUAGE plpgsql;
```

3.6.2 Output

```
select from class_cse();
select * from class_cse;
```



```
asdlab=# CREATE  SEQUENCE csekey
asdlab-# START 101;
CREATE SEQUENCE
asdlab=#
asdlab=# CREATE or REPLACE FUNCTION class_cse() RETURNS VOID as
asdlab-# $$
asdlab$# DECLARE
asdlab$# data1 VARCHAR []= '{ 0482-239091,0484-234562 ,0485-11234,0489-43617,0481-23145}';
asdlab$#  data2  VARCHAR(20)[]='{ ARUN,AMAL,PETER,JOSE,ANNIE }';
asdlab$# i INT ;
asdlab$# j INT;
asdlab$# BEGIN
asdlab$#  FOR i IN 1..5 LOOP
asdlab$#   INSERT INTO class_cse VALUES(nextval('csekey'),data2[i],data1[i]);
asdlab$#  END LOOP;
asdlab$# END;
asdlab$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
asdlab=# select class_cse();
 class_cse
-----------

(1 row)

asdlab=# select * from class_cse;
 roll | name  |    phone
------+-------+-------------
  101 | ARUN  | 0482-239091
  102 | AMAL  | 0484-234562
  103 | PETER | 0485-11234
  104 | JOSE  | 0489-43617
  105 | ANNIE | 0481-23145
(5 rows)

asdlab=#
```

Figure 7: table class cse

4 Result

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