

# 西安交通大学实验报告

课程名称：数据库基础与应用  
学院：物理学院  
班级：物试 2201  
学号：2223711803

实验名称：存储过程  
实验日期：2023.4.15  
姓名：汪洋

## 1 实验目的

1. 理解存储过程和函数的概念。
2. 掌握一种 DBMS 中存储过程的使用方法。
3. 能将存储过程应用到数据的管理中。

## 2 实验任务

练习存储过程和函数的使用。

## 3 实验环境

1. 华为云服务器
2. openGauss 数据库
3. Windows 操作系统
4. Putty

## 4 实验步骤与结果

1. 创建实验用数据表并插入 5 条实验数据，包含自己的名字。

```
1 create table student(number varchar(10), name varchar(20), birthday date);
2 insert into student values('001', '汪洋', '2004-9-5');
3 insert into student values('002', '张三', '2004-1-1');
4 insert into student values('003', '李四', '2004-2-2');
5 insert into student values('004', '王五', '2004-3-3');
6 insert into student values('005', '赵六', '2004-4-4');
```

2. 编写函数，计算两个数的和，验证该函数。

```
1 CREATE OR REPLACE FUNCTION add_numbers(a INTEGER, b INTEGER)
2 RETURNS INTEGER
3 LANGUAGE plpgsql
4 AS $$
5 BEGIN
6     RETURN a+b;
7 END;
8 $$;
9 SELECT add_numbers(2,3);
```

```

omm@ecs-5138:~
stu=# CREATE OR REPLACE FUNCTION add_numbers(a INTEGER, b INTEGER)
stu=# RETURNS INTEGER
stu=# LANGUAGE plpgsql
stu=# AS $$
stu$# BEGIN
stu$#     RETURN a+b;
stu$# END;
stu$# $$;
CREATE FUNCTION
stu=# SELECT add_numbers(2,3);
  add_numbers
-----
           5
(1 row)

stu=#

```

3. 编写函数，根据学号删除学生信息。输入参数为学号。验证该函数。

```

1 CREATE OR REPLACE FUNCTION del(p_number varchar(10))
2 RETURNS VOID
3 LANGUAGE plpgsql
4 AS $$
5 BEGIN
6     delete from student where number=p_number;
7 END;
8 $$;
9 select del('005');
10 select * from student;

```

```

omm@ecs-5138:~
stu=# CREATE OR REPLACE FUNCTION del(p_number varchar(10))
stu=# RETURNS VOID
stu=# LANGUAGE plpgsql
stu=# AS $$
stu$# BEGIN
stu$#     delete from student where number=p_number;
stu$# END;
stu$# $$;
CREATE FUNCTION
stu=# select del('005');
  del
-----
(1 row)

stu=# select * from student;
 number | name |      birthday
-----+-----+-----
  001   | 汪洋 | 2004-09-05 00:00:00
  002   | 张三 | 2004-01-01 00:00:00
  003   | 李四 | 2004-02-02 00:00:00
  004   | 王五 | 2004-03-03 00:00:00
(4 rows)

stu=#

```

4. 编写函数，插入学生信息。输入参数为学号、姓名和出生日期。验证该函数。

```

1 CREATE OR REPLACE FUNCTION insert_data(p_number varchar(10), p_name varchar(20), p_birthday
  date)
2 RETURNS VOID
3 LANGUAGE plpgsql
4 AS $$
5 BEGIN
6     insert into student values(p_number, p_name, p_birthday);
7 END;
8 $$;
9 select insert_data('005', '赵六', '2004-4-4');
10 select * from student;

```

```

omm@ecs-5138:~
stu=# CREATE OR REPLACE FUNCTION insert_data(p_number varchar(10), p_name varchar(20), p_birthday date)
stu-# RETURNS VOID
stu-# LANGUAGE plpgsql
stu-# AS $$
stu$# BEGIN
stu$#     insert into student values(p_number, p_name, p_birthday);
stu$# END;
stu$# $$;
CREATE FUNCTION
stu=# select insert_data('005', '赵六', '2004-4-4');
insert_data
-----
(1 row)

stu=# select * from student;
 number | name |      birthday
-----+-----+-----
 001    | 汪洋 | 2004-09-05 00:00:00
 002    | 张三 | 2004-01-01 00:00:00
 003    | 李四 | 2004-02-02 00:00:00
 004    | 王五 | 2004-03-03 00:00:00
 005    | 赵六 | 2004-04-04 00:00:00
(5 rows)

stu=#

```

5. 定义一个计算年龄的函数，并将其应用到学生信息的查询中，例如学生表中有“出生日期”字段 birth，日期型数据（含年月日），查询如下：

```

1 select numb, name, mycalage(birth) from student;

```

显示学生的学号、姓名和年龄。验证该函数。

```

1 CREATE OR REPLACE FUNCTION mycalage(p_birthday date)
2 RETURNS INTEGER
3 LANGUAGE plpgsql
4 AS $$
5 DECLARE
6     age INTEGER;
7 BEGIN
8     age := EXTRACT(YEAR FROM age(p_birthday));
9     RETURN age;
10 END;
11 $$;
12 select number, name, mycalage(birthday) from student;

```

```

omm@ecs-5138:~
stu=# CREATE OR REPLACE FUNCTION mycalage(p_birthday date)
stu-# RETURNS INTEGER
stu-# LANGUAGE plpgsql
stu-# AS $$
stu$# DECLARE
stu$#     age INTEGER;
stu$# BEGIN
stu$#     age := EXTRACT(YEAR FROM age(p_birthday));
stu$#     RETURN age;
stu$# END;
stu$# $$;
CREATE FUNCTION
stu=# select number, name, mycalage(birthday) from student;
 number | name | mycalage
-----+-----+-----
 001    | 汪洋 |        18
 002    | 张三 |        19
 003    | 李四 |        19
 004    | 王五 |        19
 005    | 赵六 |        19
(5 rows)

stu=#

```

6. 查看所定义的函数。

```

1 SELECT proname, proowner, pronamespace FROM pg_proc p
2 JOIN pg_namespace n ON p.pronamespace = n.oid
3 WHERE n.nspname = 'public';

```

```

omm@ecs-5138:~
stu=# SELECT proname, proowner, pronamespace FROM pg_proc p
stu=# JOIN pg_namespace n ON p.pronamespace = n.oid
stu=# WHERE n.nspname = 'public';
  proname   | proowner | pronamespace
-----+-----+-----
add_numbers |        10 |          2200
del         |        10 |          2200
insert_data |        10 |          2200
mycalage   |        10 |          2200
(4 rows)
stu=#

```

## 7. 自己做的其他函数练习。

### (1) 计算 $n!$

```

1 #使用FOR循环
2 CREATE OR REPLACE FUNCTION factorial(n INTEGER)
3 RETURNS INTEGER
4 LANGUAGE plpgsql
5 AS $$
6 DECLARE
7     result INTEGER := 1;
8     i INTEGER;
9 BEGIN
10    IF n < 0 THEN
11        RAISE EXCEPTION 'Input must be a non-negative integer.';
12    END IF;
13    FOR i IN 1..n
14    LOOP
15        result := result * i;
16    END LOOP;
17    RETURN result;
18 END;
19 $$;
20
21 #使用WHILE循环
22 CREATE OR REPLACE FUNCTION factorial(n INTEGER)
23 RETURNS INTEGER
24 LANGUAGE plpgsql
25 AS $$
26 DECLARE
27     result INTEGER := 1;
28     i INTEGER := 1;
29 BEGIN
30    IF n < 0 THEN
31        RAISE EXCEPTION 'Input must be a non-negative integer.';
32    END IF;
33    WHILE i <= n
34    LOOP
35        result := result * i;
36        i := i + 1;
37    END LOOP;
38    RETURN result;
39 END;
40 $$;

```

### (2) 计算斐波那契数列第 $n$ 项

```

1 #循环解法
2 CREATE OR REPLACE FUNCTION fibonacci(n INTEGER)
3 RETURNS INTEGER

```

```

4 LANGUAGE plpgsql
5 AS $$
6 DECLARE
7     a INTEGER := 0;
8     b INTEGER := 1;
9     i INTEGER;
10    temp INTEGER;
11 BEGIN
12     IF n <= 1 THEN
13         RETURN n;
14     END IF;
15     FOR i IN 2..n LOOP
16         temp := a + b;
17         a := b;
18         b := temp;
19     END LOOP;
20     RETURN b;
21 END;
22 $$;
23
24 #递归解法
25 CREATE OR REPLACE FUNCTION fibonacc(i INTEGER)
26 RETURNS INTEGER
27 LANGUAGE plpgsql
28 AS $$
29 BEGIN
30     IF n <= 1 THEN
31         RETURN n;
32     ELSE
33         RETURN fibonacc(n - 1) + fibonacc(n - 2);
34     END IF;
35 END;
36 $$;

```

### (3) 计算 10 以内的奇数和

```

1 #使用FOR循环
2 CREATE OR REPLACE FUNCTION sum_odds(n INTEGER)
3 RETURNS INTEGER
4 LANGUAGE plpgsql
5 AS $$
6 DECLARE
7     sum_odds INTEGER := 0 ;
8     i INTEGER;
9 BEGIN
10    FOR i IN 1..10
11    LOOP
12        IF i % 2 != 0 THEN
13            sum_odds := sum_odds + i;
14        END IF;
15    END LOOP;
16    RETURN sum_odds;
17 END;
18 $$;
19
20 #使用WHILE循环
21 CREATE OR REPLACE FUNCTION sum_odds(n INTEGER)
22 RETURNS INTEGER
23 LANGUAGE plpgsql
24 AS $$
25 DECLARE
26     sum_odds INTEGER := 0 ;
27     i INTEGER := 1;
28 BEGIN

```

```

29     WHILE 1 LOOP #此处1被视为真，创建了一个无限循环
30         IF i >= 10 THEN
31             EXIT;
32         END IF;
33         IF i % 2 != 0 THEN
34             sum_odds := sum_odds + i;
35             i = i + 1;
36             CONTINUE;
37         END IF;
38         i = i + 1;
39     END LOOP;
40     RETURN sum_odds;
41 END;
42 $$;

```

8. 删除自定义的函数和数据表。

```

1 drop function add_numbers;
2 drop function del;
3 drop function insert_data;
4 drop function mycalage;
5 drop table student;

```

## 5 实验总结

1. 总体来说非常简单。
2. 当函数参数与列名重复的时候很容易报错，尽量要使用不重复的名称。
3. 在 FOR 循环中， $i := i + 1$  不需显式地写出；在 WHILE 循环中， $i := i + 1$  需要显式地写出。