

实验十二 PL/pgSQL程序

实验目的

- 了解PL/SQL程序的结构
- 掌握简单的PL/SQL程序的编写
- 掌握PL/SQL程序的调用方法

实验内容

- 编写简单的PL/SQL函数，输入参数、定义变量、输出相应结果
- 编写PL/SQL函数，通过函数对数据表进行修改
- 调用已编写的相应程序

实验思路

- 创建一个函数，输入字符串'database management system'，输出'management'；

思路：

根据实验提示，选择使用SQL中的 `substring` 函数；

同时为了提高代码的复用性，定义函数 `substr(a varchar, b int, c int)` ,含义与 `substring` 中参数的意义相同，用于划分字符串；

因此声明一个字符串 `ans` ，用于表示最后的结果。

使用

```
SELECT substring(a,b,c) INTO ans;
```

将结果保存至ans中，最后返回ans，即为输出结果；

- 创建一个函数，使用函数调用的方法，将teacher表中所有教师工资加200，并输出教师人数；

teacher表如图所示：

	teacher_id [PK] integer	teacher_name character varying (20)	teacher_salary integer	teacher_course character varying
1	202201	Zang	4600	Math
2	202202	Wei	4200	English
3	202203	Yuan	5700	Chemistry
4	202204	Zhao	3900	History
5	202205	Li	5400	Physics
6	202206	Zhang	4300	PE

思路：

依然以代码具有 **普适性**为目的，定义函数 `addsalary(money int)` ，其中 `money` 变量为增加的薪水数量，函数返回值为 `int` ，代表教师人数。

主要思路是解决一个问题——**如何遍历表中某一列的数值？**

最一开始的思路认为，在数据库中挑选某一行是并不容易的，必须用到 `WHERE` 语句，然而想要正确选择某一行需要用到其主键（主键具备唯一性），而如果主键只知道类型而不知道具体数值，直接使用循环加主键去查找会比较麻烦(可能不具备规律)，初想法是想像高级语言一样去“遍历”主键，但是对于不同的表其主键可能并不会具备一定的规律性；

为了解决循环的问题，选择使用 ORDER BY 语句对表按照主键进行排序，由于主键的唯一性，因此排序是唯一的，不会产生实验四中提及到的“使用LIMIT的结果可能和ORDER BY的结果不同”的情况。则挑选第i行可以写作 limit 1 offser i，即完成了 顺序选择某一行 的要求；而为了选取正确的行用来修改 update，而 update 语句需要用到 WHERE 语句去定位，因此选择再定义一个与主键类型相同的变量用来记录当前行的主键，以进行对表的更新。

最终返回教师人数 cnt。

实验结果

- substr 函数结果如下：

- 运行代码

```
SELECT substr('database management system',10 ,10);
```

- 运行结果

	substr text	
1	management	

- addsalary() 函数：

- 运行代码

```
SELECT addsalary(200);
```

- 运行结果如下：

	addsalary integer	
1	6	

使用pgsql中的输出函数 raise 打印函数运行中的teacher_salary变化后的结果，输出如下；

Data output	消息	通知
NOTICE:	4800	
NOTICE:	4400	
NOTICE:	5900	
NOTICE:	4100	
NOTICE:	5600	
NOTICE:	4500	

调用语句，运行结果如下，和输出值一样，与一开始的表相比确实每个老师的薪资都增加了200，说明程序可行。

```
SELECT * FROM public.teacher
ORDER BY teacher_id ASC
```

	teacher_id [PK] integer	teacher_name character varying (20)	teacher_salary integer	teacher_course character varying
1	202201	Zang	4800	Math
2	202202	Wei	4400	English
3	202203	Yuan	5900	Chemistry
4	202204	Zhao	4100	History
5	202205	Li	5600	Physics
6	202206	Zhang	4500	PE

源程序

substr(a varchar, b int, c int) 函数，参数含义与 substring() 函数一致

```
CREATE OR REPLACE FUNCTION Substr(a varchar,b int,c int)  --创建函数，确定参数类型
returns varchar  --返回值类型
as
$$
declare ans VARCHAR;  --声明变量，结果存储到里面
BEGIN
    SELECT substring(a,b,c) INTO ans;  --使用substring函数，将结果保存至ans中
    return ans;  --返回结果
END;
$$
LANGUAGE plpgsql;
```

addsalary(money int) 函数，参数为增加的薪酬数目

```
CREATE OR REPLACE FUNCTION addsalary(money int)  --函数名称
RETURNS INT  --返回教师人数
AS
$$
DECLARE i int;  --循环下标
DECLARE cnt int;  --人数
DECLARE num int;  --表中的主键，即记录teacher_id值
DECLARE now_salary int;  --当前老师的薪资
BEGIN
    i:=0;  --初始化
    cnt:=0;  --初始化
    select count(*) into cnt from teacher;  --记录教师人数
    while i<cnt  --循环开始
    loop
        select teacher_salary into now_salary from teacher order by teacher_id asc limit 1 offset i;  --选择按照teacher_id升序排序的第i个人的薪资
        select teacher_id into num from teacher order by teacher_id asc limit 1 offset i;  --记录当前的主键值，即第i个人的teacher_id
        now_salary:=now_salary+money;  --加钱！
        update teacher set teacher_salary=now_salary where teacher_id=num;  --更新薪水
        raise notice '%', now_salary;  --输出，可不要，用于debug
        i:=i+1;  --下标加1
    end loop;
    return cnt;  --返回教师人数
end;
$$ LANGUAGE plpgsql;
```

实验体会

- SQL中的 `substring(a varchar,b int,c int)` 函数获取在字符串a中从b开始的c个字符构成的子串；SQL不同于C等高级语言，其字符串是从1开始编号的；
- 对于循环来说，除了使用上面的方法，还可以使用游标的方式；
- pgsql的函数写法与高级语言既有相似之处：声明变量名、定义变量类型、主函数；不同之处在于会首先写出返回的类型，并且先写变量名称、再写数据类型，过程中还可以使用sql语言获得某个变量值，比较方便；
- 存储函数书写虽然有一定难度（不太熟练），但是使用起来非常方便，可以节省很多机械性的操作所产生的时间耗费，整个体系更加便利、富有逻辑化。

思考题

- 使用函数调用和直接使用SQL对数据表进行修改有何区别？

使用函数调用对表进行修改的话，整个实现比较简单，写好函数后只需要写一条执行语句即可，方便快捷，效率高，且函数可以支持参数，并且能够设置返回类型，更加智能，适用范围广，可以应用于大量数据的修改；同时定义自定义函数可以让数据库自己计算，减少与高级语言之间的交互时间，提高效率。

直接使用SQL对数据表进行修改的话比较繁琐，对于修改语句 `update` 来说，虽然有批量处理的方法，但是性能比较差，且书写起来冗杂麻烦，因此不适合大范围大量的数据改动，但是对于部分零散的数据修改而言，简单方便。