



## 数据库引航 第9课—PL/SQL 存储过程和函数

2012.10.15

**【声明】** 本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

<http://edu.dataguru.cn>

# PL/SQL--- Procedure Language



- ◆ 匿名PL/SQL块
  - 动态构造，只能执行一次
- ◆ 存储过程，函数，触发器
  - 存储在数据库中编译过的代码，可以反复执行。

2012.10.15

## 从最简单的PL/SQL开始

```
SQL> begin
  2  for i in 1..10 loop
  3  null;
  4  end loop;
  5  end;
  6  /
```

PL/SQL 过程已成功完成。

```
SQL>
```

# Hello,world!

```
SQL> set serveroutput on;
SQL> ed
已写入 file afiedt.buf

  1  begin
  2  for i in 1..10 loop
  3  dbms_output.put_line('hello,world!');
  4  end loop;
  5* end;
SQL>
SQL> /
hello,world!
hello,world!
hello,world!
hello,world!
hello,world!
hello,world!
hello,world!
hello,world!
hello,world!
hello,world!

PL/SQL 过程已成功完成。
```

2012.10.15

## 循环插入

```
SQL> create table t (id int);  
表已创建。  
  
SQL> begin  
  2  for i in 1..100 loop  
  3  insert into t values(i) ;  
  4  end loop;  
  5  end;  
  6  /  
  
PL/SQL 过程已成功完成。  
  
SQL> commit;  
提交完成。  
  
SQL> select count(*) from t;  
  
  COUNT(*)  
-----  
        100  
  
SQL>
```

2012.10.15

## 定义变量的匿名PL/SQL块

```
1  declare
2  x varchar2(40):='my first PL/SQL block';
3  begin
4  dbms_output.put_line(x);
5* end;
SQL> /
my first PL/SQL block

PL/SQL 过程已成功完成。
```

## 游标-cursor



游标是一个指向上下文的句柄( handle)或指针。通过游标，PL/SQL可以控制上下文区和处理语句时上下文区会发生什么事情。

```
1  declare
2  x t.id%type;
3  cursor c is select * from t;
4  begin
5  open c;
6  loop
7  fetch c into x;
8  exit when c%notfound;
9  dbms_output.put_line('id is '||x);
10 end loop;
11 close c;
12* end;
SQL> /
id is 90
id is 91
id is 92
id is 93
id is 94
id is 95
id is 96
id is 97
id is 98
id is 99
id is 100

PL/SQL 过程已成功完成。
```

2012.10.15



- ◆ 编译好的PL/SQL块，有自己的名称，保存在数据库中可以被调用执行。
- ◆ 可以输入参数

```
CREATE [OR REPLACE] PROCEDURE Procedure_name
    [ (argument [ { IN | IN OUT } ] Type,
      argument [ { IN | OUT | IN OUT } ] Type ]
    { IS | AS }
    <类型.变量的说明>
BEGIN
    <执行部分>
EXCEPTION
    <可选的异常错误处理程序>
END;
```

# 一个简单的存储过程



## ◆ 删除指定员工记录

```
CREATE OR REPLACE PROCEDURE DelEmp(v_empno IN emp.empno%TYPE) AS
    No_result EXCEPTION;
BEGIN
    DELETE FROM emp WHERE empno=v_empno;
    IF SQL%NOTFOUND THEN
        RAISE no_result;
    END IF;
    DBMS_OUTPUT.PUT_LINE('编码为'||v_empno||'的员工已被除名!');
EXCEPTION
    WHEN no_result THEN
        DBMS_OUTPUT.PUT_LINE('你需要的数据不存在!');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('发生其它错误!');
END DelEmp;
```

2012.10.15

## 存储过程的参数 --IN,OUT,IN OUT

```
CREATE OR REPLACE PROCEDURE ModeTest (  
    p_InParameter IN NUMBER,  
    p_OutParameter OUT NUMBER,  
    p_InOutParameter IN OUT NUMBER) IS  
    v_LocalVariable NUMBER;  
  
BEGIN  
    v_LocalVariable := p_InParameter; -- Legal  
    p_InParameter := 7; -- Illegal  
    p_OutParameter := 7; -- Legal  
    v_LocalVariable := p_outParameter; -- Illegal  
    v_LocalVariable := p_InOutParameter; -- Legal  
    p_InOutParameter := 7; -- Legal  
END ModeTest;  
/
```

2012.10.15

```
1 create or replace procedure drop_table is
2 begin
3 drop table t purge;
4* end;
SQL> /
```

警告：创建的过程带有编译错误。

```
SQL> show error;
PROCEDURE DROP_TABLE 出现错误:
```

LINE/COL ERROR

```
-----
3/1      PLS-00103: 出现符号 "DROP"在需要下列之一时:
          < begin case declare exit
          for goto if loop mod null pragma raise return select update
          while with <an identifier>
          <a double-quoted delimited-identifier> <a bind variable> <<
          continue close current delete fetch lock insert open rollback
          savepoint set sql execute commit forall merge pipe purge
```

```
SQL> create or replace procedure drop_table is
  2  begin
  3  execute immediate 'drop table t purge';
  4  end;
  5  /
```

过程已创建。

```
SQL> exec drop_table;
```

PL/SQL 过程已成功完成。

```
SQL> desc t;
```

ERROR:

ORA-04043: 对象 t 不存在

```
SQL>
```

```
SQL> create or replace procedure drop_table is
  2  begin
  3  execute immediate 'droping table t purge';
  4  end;
  5  /
```

过程已创建。

```
SQL> exec drop_table;
BEGIN drop_table; END;
```

✖

第 1 行出现错误:

ORA-00900: 无效 SQL 语句

ORA-06512: 在 "TEST.DROP\_TABLE", line 3

ORA-06512: 在 line 1

- ◆ 函数要返回一个结果。
- ◆ 函数可以在SQL语句中调用。

```
CREATE [OR REPLACE] FUNCTION function_name
    [(argument [ { IN| IN OUT } ] type,
    argument [ { IN | OUT | IN OUT } ] type]
    RETURN return_type
    { IS | AS }
    <类型.变量的说明>
BEGIN
    FUNCTION_body
EXCEPTION
    其它语句
END;
```



## 一个简单的函数--加法

```
SQL> ed
```

```
已写入 file afiedt.buf
```

```
1 create or replace function mysum(a in number,b in number)
2 return number is
3 v_sum number;
4 begin
5 v_sum:=a+b;
6 return v_sum;
7* end mysum;
```

```
SQL> /
```

函数已创建。

```
SQL> select mysum(1,2) from dual;
```

```
MYSUM(1,2)
```

```
-----
```

```
3
```

2012.10.15

- Dataguru (炼数成金) 是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 <http://edu.dataguru.cn>



# Thanks

**FAQ时间**