

讲师: collen7788@126.com

P r e s e n t a t i o n

存储过程，函数和触发器

本章目标

1

掌握存储过程

2

掌握存储函数

3

掌握触发器

存储过程和存储函数

- ❖ 指存储在数据库中供所有用户程序调用的子程序叫存储过程、存储函数。

创建存储过程

❖ 用**CREATE PROCEDURE**命令建立存储过程。

❖ 语法：

create [or replace] PROCEDURE 过程名(参数列表)

AS

PLSQL子程序体;

存储过程示例

- ❖ 为指定的职工在原工资的基础上长**10%**的工资，并打印涨工资前和涨工资后的工资

```
1  /*
2  为指定的职工在原工资的基础上长10%的工资,并打印涨工资前和涨工资后的工资
3
4  可能用到的sql语句
5  update emp set sal = sal * 1.1 where empno = empid;
6
7  */
8
9  create or replace procedure raiseSalary(empid in number)
10 as
11     pSal emp.sal%type; --保存员工当前工资
12 begin
13     --查询该员工的工资
14     select sal into pSal from emp where empno=empid;
15     --给该员工涨工资
16     update emp set sal = sal * 1.1 where empno = empid;
17
18     --打印涨工资前后的工资
19     dbms_output.put_line('员工号:' || empid || ' 涨工资前:' || pSal || ' 涨工资后' || pSal * 1.1);
20 end;
21 /
```

存储过程调用

❖ 方法一:

```
set serveroutput on  
begin  
    raisesalary(7369);  
end;  
/
```

❖ 方法二:

```
set serveroutput on  
exec raisesalary(7369);
```

练习：为指定员工增加指定额度的工资(传递多个参数)

```
create or replace procedure raiseSalary2(empid in number, rate in NUMBER)
as
  pSal emp.sal%type; --保存员工当前工资
begin
  --查询该员工的工资
  select sal into pSal from emp where empno=empid;
  --给该员工涨工资
  update emp set sal = sal * rate where empno = empid;

  --打印涨工资前后的工资
  dbms_output.put_line('员工号:' || empid || ' 涨工资前:' || pSal || ' 涨工资后' || pSal * rate);
end;
/
```

存储函数

❖ 函数（**Function**）为一命名的存储程序，可带参数，并返回一计算值。函数和过程的结构类似，但必须有一个**RETURN**子句，用于返回函数值。函数说明要指定函数名、结果值的类型，以及参数类型等。

❖ 建立存储函数的语法：

```
CREATE [OR REPLACE] FUNCTION 函数名(参数列表)  
RETURN 函数值类型  
AS  
    PLSQL子程序体;
```


示例：查询某职工的年收入。

```
1  /*
2  查询某职工的总收入。
3  */
4
5  create or replace function queryEmpSalary(empid in number)
6      RETURN NUMBER
7  as
8      pSal number; --定义变量保存员工的工资
9      pComm number; --定义变量保存员工的奖金
10 begin
11     select sal,comm into pSal, pcomm from emp where empno = empid;
12     return pSal*12+ nvl(pcomm,0);
13 end;
14 /
```

函数的调用

```
declare
    v_sal number;
begin
    v_sal:=queryEmpSalary(7934);
    dbms_output.put_line('salary is:' || v_sal);
end;
/
```

```
begin
    dbms_output.put_line('salary is:' || queryEmpSalary(7934));
end;
```

过程和函数中的**in**和**out**

- ❖ 一般来讲，过程和函数的区别在于函数可以有一个返回值；而过程没有返回值。
- ❖ 但过程和函数都可以通过**out**指定一个或多个输出参数。我们可以利用**out**参数，在过程和函数中实现返回多个值。

什么时候用存储过程/存储函数？

❖ 原则：

- 如果只有一个返回值，用存储函数；否则，就用存储过程。

在Java语言中调用存储过程

- 存储过程:

```
1 create or replace procedure testprocedure(eid in number, empname out varchar,empsal out NUMBER )
2 as
3 begin
4   select ename,sal into empname, empsal from emp where empno=eid;
5 end;
6
```

- Java程序:

```
//创建CallableStatement
CallableStatement call = conn.prepareCall("{call testprocedure(?,?,?)}");
//设置参数
call.setInt(1, 7369);
call.registerOutParameter(2, oracle.jdbc.OracleTypes.VARCHAR);
call.registerOutParameter(3, oracle.jdbc.OracleTypes.NUMBER);
//执行存储过程
call.execute();
//输出结果
String name = call.getString(2);
int sal = call.getInt(3);
System.out.println(name);
System.out.println(sal);
```

在Java语言中调用存储函数

- 存储函数:

```
1 create or replace function testfunction(eid in number, empname out varchar, empsal out NUMBER )
2 return NUMBER
3 as
4 begin
5     select ename, sal into empname, empsal from emp where empno=eid;
6     return empsal * 12;
7 end;
```

- Java程序

```
//创建CallableStatement
CallableStatement call = conn.prepareCall("(?=call testfunction(?, ?, ?))");
//设置参数
call.registerOutParameter(1, oracle.jdbc.OracleTypes.NUMBER);
call.setInt(2, 7369);
call.registerOutParameter(3, oracle.jdbc.OracleTypes.VARCHAR);
call.registerOutParameter(4, oracle.jdbc.OracleTypes.NUMBER);
//执行存储函数
call.execute();
//输出结果
int annualSal = call.getInt(1);
String name = call.getString(3);
int sal = call.getInt(4);
System.out.println(annualSal);
System.out.println(name);
System.out.println(sal);
```

在out参数中使用游标

❖ 问题：查询某个部门中所有员工的所有信息

❖ 申明包结构

```
CREATE OR REPLACE PACKAGE MYPACKAGE AS  
    type empcursor is ref cursor;  
    procedure queryEmpList(dno in number, empList out empcursor);  
END MYPACKAGE;
```

❖ 创建包体

```
CREATE OR REPLACE PACKAGE BODY MYPACKAGE AS  
  
    procedure queryEmpList(dno in number, empList out empcursor) AS  
    BEGIN  
        open empList for select * from emp where deptno=dno;  
    END queryEmpList;  
  
END MYPACKAGE;
```

在Java语言中访问游标类型的out参数

- 创建Statement

```
CallableStatement call = conn.prepareCall("{call mypackage.queryemp(?,?)}");
```

- 设置参数

```
//第一个参数: 要查询的员工编号 in  
//第二个参数: 员工的信息 游标 out  
call.setInt(1, 7369);  
call.registerOutParameter(2, oracle.jdbc.OracleTypes.CURSOR);
```

- 执行

```
//执行调用  
call.execute();
```

- 输出结果

```
//输出结果  
ResultSet rs = ((OracleCallableStatement) call).getCursor(2);  
while(rs.next()){  
    System.out.println(rs.getInt(1) + " " + rs.getString(2) + " " + rs.getString(3));  
}
```


触发器

- ❖ 数据库触发器是一个与表相关联的、存储的**PL/SQL**程序。每当一个特定的数据操作语句(**Insert,update,delete**)在指定的表上发出时,**Oracle**自动地执行触发器中定义的语句序列。

- ❖ 触发器的类型

- 语句级触发器
 - 在指定的操作语句操作之前或之后执行一次，不管这条语句影响了多少行。
- 行级触发器（**FOR EACH ROW**）
 - 触发语句作用的每一条记录都被触发。在行级触发器中使用:**old**和:**new**伪记录变量, 识别值的状态。

创建触发器

```
CREATE [or REPLACE] TRIGGER 触发器名  
{BEFORE | AFTER}  
{DELETE | INSERT | UPDATE [OF 列名]}  
ON 表名  
[FOR EACH ROW [WHEN(条件) ]]  
PLSQL 块
```

示例1：限制非工作时间向数据库插入数据

```
1 create or replace
2 trigger securityEmp
3 before insert on emp
4 declare
5
6 begin
7
8 if to_char(sysdate,'day') in ('星期四','星期六','星期天')
9 or to_number(to_char(sysdate,'hh24')) not between 8 and 18 then
10 raise_application_error(-20001,'不能在非工作时间插入数据. ');
11 end if;
12 end;
```

-20000到-20999之间

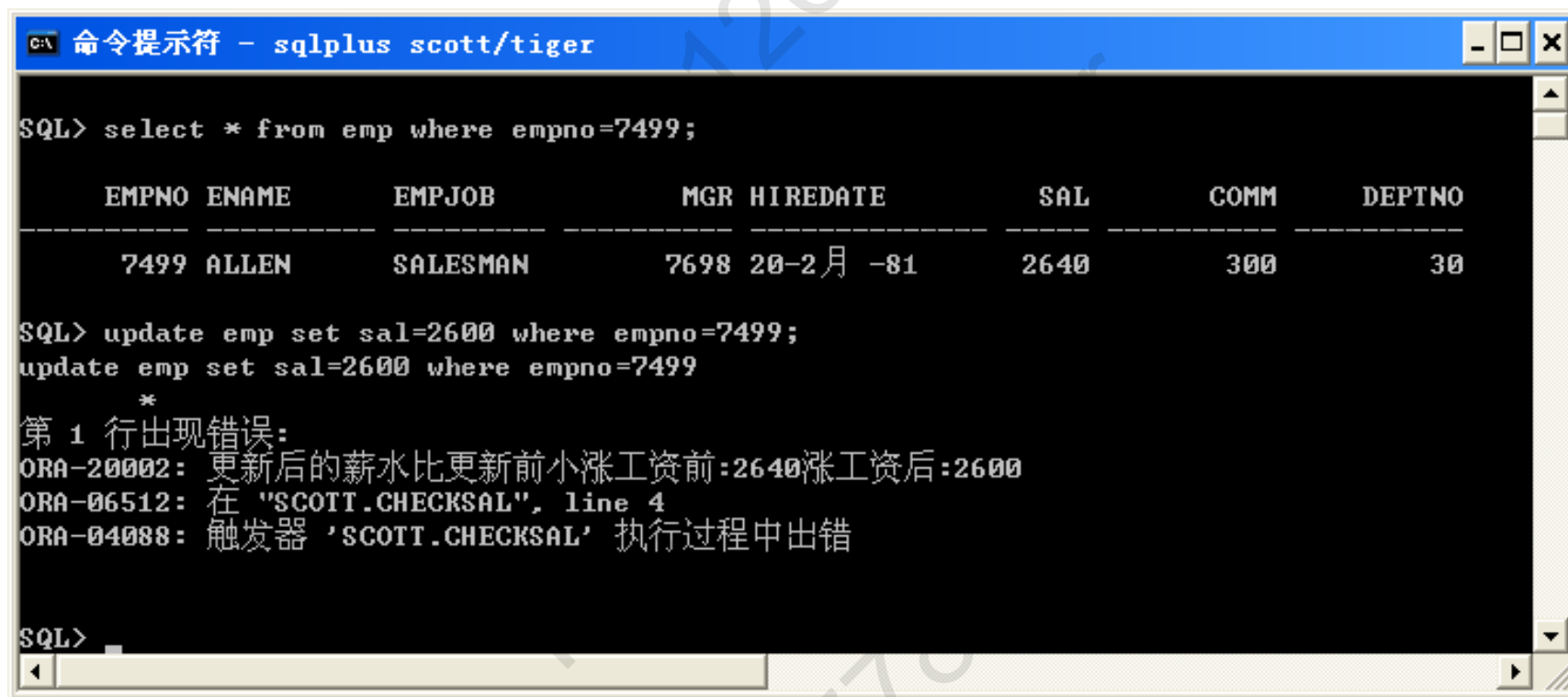
触发语句与伪记录变量的值

触发语句	:old	:new
Insert	所有字段都是空(null)	将要插入的数据
Update	更新以前该行的值	更新后的值
delete	删除以前该行的值	所有字段都是空(null)

示例2： 确认数据（检查emp表中sal 的修改值不低于原值）

```
1  /*
2  示例2：确认数据（检查emp表中sal 的修改值不低于原值）
3  */
4
5  create or replace trigger checkSal
6  before update of sal on emp
7  for each row
8  declare
9  begin
10     if :new.sal <:old.sal THEN
11         raise_application_error(-20002, '更新后的薪水比更新前小');
12     end if;
13 end;
```

运行效果:



```
C:\ 命令提示符 - sqlplus scott/tiger

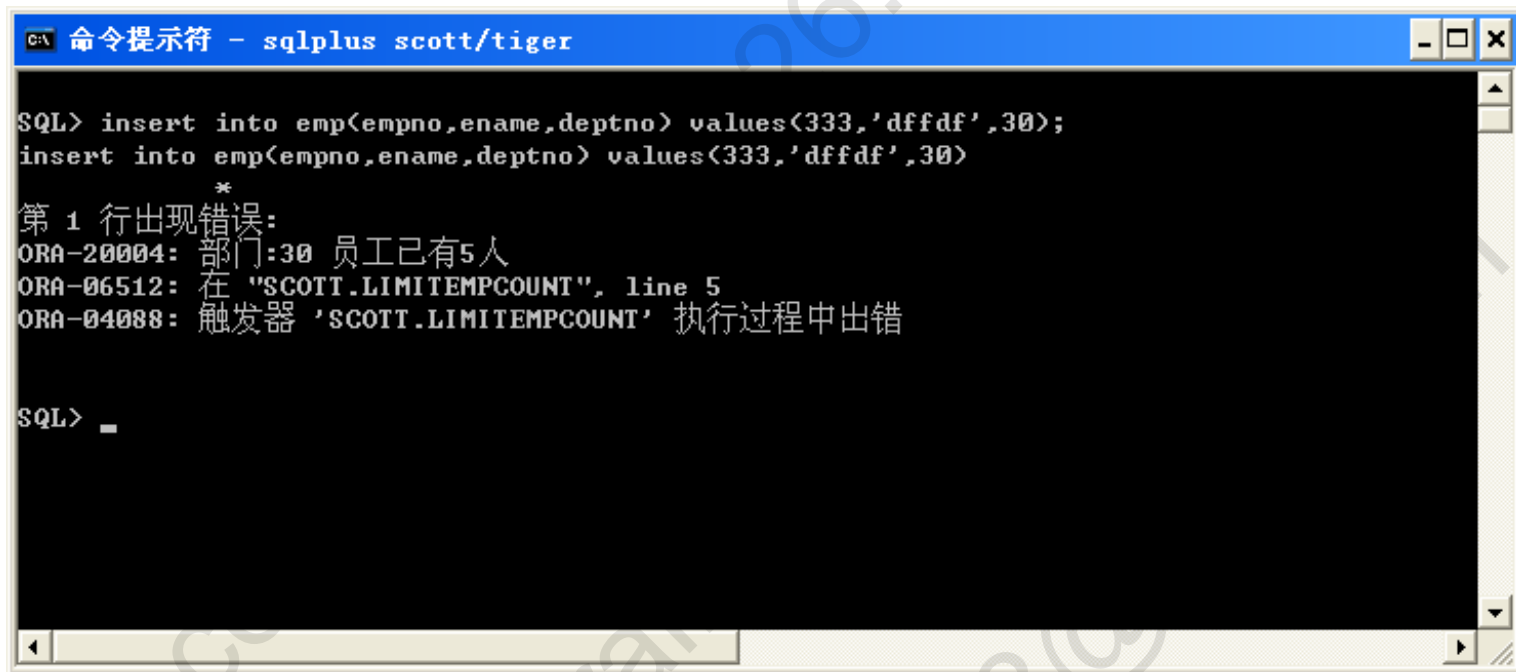
SQL> select * from emp where empno=7499;

   EMPNO ENAME      EMPJOB      MGR HIREDATE          SAL          COMM          DEPTNO
-----
   7499 ALLEN      SALESMAN      7698 20-2月 -81      2640          300           30

SQL> update emp set sal=2600 where empno=7499;
update emp set sal=2600 where empno=7499
      *
第 1 行出现错误:
ORA-20002: 更新后的薪水比更新前小涨工资前:2640涨工资后:2600
ORA-06512: 在 "SCOTT.CHECKSAL", line 4
ORA-04088: 触发器 'SCOTT.CHECKSAL' 执行过程中出错

SQL>
```

练习： 限制每个部门只招聘**5**名职工，超过计划则报出错误信息



A screenshot of a Windows command prompt window titled "命令提示符 - sqlplus scott/tiger". The window has a blue title bar and standard Windows window controls (minimize, maximize, close). The background is black with white text. The text shows two SQL insert statements, followed by an asterisk and a message indicating an error on the first line. Three Oracle error messages are displayed: ORA-20004, ORA-06512, and ORA-04088. The prompt "SQL>" is visible at the bottom.

```
C:\ 命令提示符 - sqlplus scott/tiger

SQL> insert into emp(empno,ename,deptno) values(333,'dffdf',30);
insert into emp(empno,ename,deptno) values(333,'dffdf',30)
*
第 1 行出现错误:
ORA-20004: 部门:30 员工已有5人
ORA-06512: 在 "SCOTT.LIMITEMPCOUNT", line 5
ORA-04088: 触发器 'SCOTT.LIMITEMPCOUNT' 执行过程中出错

SQL> _
```

触发器总结

❖ 触发器可用于

- 数据确认
- 实施复杂的安全性检查
- 做审计，跟踪表上所做的数据操作等
- 数据的备份和同步

❖ 查询触发器、过程及函数

- `select * from user_triggers;`
- `select * from user_source;`

讲师: collen7788@126.com

P r e s e n t a t i o n

Thank you