1 存储过程

1.1 语法结构

```
CREATE [ OR REPLACE ] PROCEDURE [ schema. ] procedure_name
  [ ( parameter_declaration [, parameter_declaration ]... ) ]
  [ invoker_rights_clause ]
  { IS | AS }
  { [ declare_section ] body | call_spec | EXTERNAL} ;
```

1.2 演示

1.2.1 没有参数的过程

```
CREATE OR REPLACE PROCEDURE proc_get_emp_info IS

  v_ename VARCHAR2(100);

  v_err VARCHAR2(100);
BEGIN
  dbms_output.put_line('EXEC:' || v_err);
  BEGIN
    SELECT ename INTO v_ename FROM emp WHERE empno = 7369;

  EXCEPTION
    WHEN OTHERS THEN
      v_err := 'err';
  END;

  IF v_err IS NOT NULL THEN
    RETURN;
  ELSE
    dbms_output.put_line('Add other deal logic....');
```

```
  END IF;

EXCEPTION
  WHEN OTHERS THEN
    dbms_output.put_line('err.....');
END proc_get_emp_info;
```

1.2.2 有参数的过程

```
CREATE OR REPLACE PROCEDURE proc_get_emp_info(pi_empno IN NUMBER,
                                              po_ename OUT VARCHAR2) IS

  v_ename VARCHAR2(100);

  v_err VARCHAR2(100);
BEGIN
  dbms_output.put_line('EXEC:' || v_err);

  BEGIN
    SELECT ename INTO v_ename FROM emp WHERE empno = pi_empno;

  EXCEPTION
    WHEN OTHERS THEN
      v_err := 'err';
  END;

  IF v_err IS NOT NULL THEN
    RETURN;
  ELSE
    dbms_output.put_line('Ename is ' || v_ename);
  END IF;

  po_ename := v_ename;

EXCEPTION
```

```
  WHEN OTHERS THEN
    dbms_output.put_line('err.....');
END proc_get_emp_info;
```

1.2.3 返回错误处理信息

```
CREATE OR REPLACE PROCEDURE proc_get_emp_info(pi_deptno IN NUMBER,
                                              po_err OUT VARCHAR2) IS

  v_ename VARCHAR2(100);
BEGIN
  for rec in(select *from emp where deptno=pi_deptno)loop
    dbms_output.put_line(rec.ename);
  end loop;

EXCEPTION
  WHEN OTHERS THEN
    po_err :='err';
    dbms_output.put_line('err.....');
END proc_get_emp_info;
```

2 函数

2.1 语法结构

```
CREATE [ OR REPLACE ] FUNCTION [ schema. ] function_name
  [ ( parameter_declaration [, parameter_declaration]... )
  ]
  RETURN datatype
  [ { invoker_rights_clause
    | DETERMINISTIC
    | parallel_enable_clause
    | RESULT_CACHE  [ relies_on_clause ]
```

```
  }...
 ]
 { { AGGREGATE | PIPELINED }  USING [ schema. ] implementation_type
 | [ PIPELINED ] { IS | AS } { [ declare_section ] body
                            | call_spec
                            | EXTERNAL
                            }
 } ;
```

2.2 演示

2.2.1 返回方式1

```
CREATE OR REPLACE FUNCTION func_get_emp_name(pi_empno IN NUMBER)
  RETURN VARCHAR2 IS

  v_name VARCHAR2(100);
BEGIN
  BEGIN
    SELECT ename INTO v_name FROM emp WHERE empno = pi_empno;

  EXCEPTION
    WHEN OTHERS THEN
      v_name := NULL;
  END;

  RETURN v_name;
END func_get_emp_name;
```

2.2.2 返回方式2

```
CREATE OR REPLACE FUNCTION func_get_emp_name(pi_empno IN NUMBER)
  RETURN VARCHAR2 IS

  v_name VARCHAR2(100);
```

```
BEGIN
  SELECT ename INTO v_name FROM emp WHERE empno = pi_empno;

  RETURN v_name;

EXCEPTION
  WHEN OTHERS THEN
    RETURN v_name;

END func_get_emp_name;
```

3 Package概述

3.1 语法结构

3.1.1 包头

```
CREATE [ OR REPLACE ] PACKAGE [ schema. ] package_name
   [ invoker_rights_clause ]
   { IS | AS } item_list_1 END [ package_name ] ;
```

3.1.2 包体

```
CREATE [ OR REPLACE ] PACKAGE BODY [ schema. ] package_name
{ IS | AS } declare_section [ initialize_section ]
END [ package_name ] ;
```

3.2 演示

3.2.1 包头

```
CREATE OR REPLACE PACKAGE pkg_emp IS
  /*****************************************\
  PKG Name :
```

```
  Function Desc :
  Author :
  Create Date :
  \****************************************/

  c_pkg CONSTANT VARCHAR2(100) := 'PKG_EMP';
  PROCEDURE proc_get_emp_info(pi_deptno IN NUMBER, po_err OUT VARCHAR2);

  FUNCTION func_get_emp_name(pi_empno IN NUMBER) RETURN VARCHAR2;

  PROCEDURE proc_log(pi_pkg IN VARCHAR2, pi_msg IN VARCHAR2);

  PROCEDURE proc_update_emp(pi_deptname IN VARCHAR2);

  PROCEDURE proc_test_auto_transc(pi_deptname IN VARCHAR2);
END pkg_emp;
```

3.2.2 包体

```
CREATE OR REPLACE PACKAGE BODY pkg_emp IS
  /****************************************\
  PKG Name :
  Function Desc :
  Author :
  Create Date :
  \****************************************/

  PROCEDURE proc_print_time IS
    v_date VARCHAR2(100);
  BEGIN
    SELECT to_char(SYSDATE, 'yyyy-mm-dd hh24:mi:ss') INTO v_date FROM dual;

    dbms_output.put_line('Current time is ' || v_date);
  END proc_print_time;
```

```
  PROCEDURE proc_get_emp_info(pi_deptno IN NUMBER, po_err OUT VARCHAR2)
IS
  BEGIN
    FOR rec IN (SELECT * FROM emp WHERE deptno = pi_deptno) LOOP
      dbms_output.put_line(rec.ename);
    END LOOP;

  EXCEPTION
    WHEN OTHERS THEN
      dbms_output.put_line('err.....');
  END proc_get_emp_info;

  FUNCTION func_get_emp_name(pi_empno IN NUMBER) RETURN VARCHAR2 IS

    v_name VARCHAR2(100);
  BEGIN

    proc_print_time;

    BEGIN
      SELECT ename INTO v_name FROM emp WHERE empno = pi_empno;

    EXCEPTION
      WHEN OTHERS THEN
        v_name := NULL;
    END;

    proc_print_time;

    RETURN v_name;
  END func_get_emp_name;

  PROCEDURE proc_log(pi_pkg IN VARCHAR2, pi_msg IN VARCHAR2) IS
    PRAGMA AUTONOMOUS_TRANSACTION;
```

```
  v_pkg VARCHAR2(100) := pi_pkg;
  v_msg VARCHAR2(100) := pi_msg;
BEGIN
  INSERT INTO t_log VALUES (seq_log.nextval, v_pkg, v_msg);

  COMMIT;
END proc_log;

PROCEDURE proc_update_emp(pi_deptname IN VARCHAR2) IS
  v_deptno NUMBER;
  v_ename  VARCHAR2(100);
  v_msg    VARCHAR2(100);
BEGIN
  BEGIN
    SELECT deptno INTO v_deptno FROM dept WHERE dname = pi_deptname;

  EXCEPTION
    WHEN OTHERS THEN
      v_deptno := NULL;
  END;

  UPDATE emp SET comm = 1000 WHERE deptno = v_deptno;

  SELECT ename INTO v_ename FROM emp WHERE deptno = v_deptno;

  COMMIT;

EXCEPTION
  WHEN OTHERS THEN
    v_msg := substr(SQLERRM, 1, 100);
    pkg_emp.proc_log(pi_pkg => c_pkg, pi_msg => v_msg);
    ROLLBACK;
END proc_update_emp;

--自治事务演示
```

```
PROCEDURE proc_test_auto_transc(pi_deptname IN VARCHAR2) IS
  v_deptno NUMBER;
  v_ename  VARCHAR2(100);
  v_msg    VARCHAR2(100);
BEGIN
  v_msg := '1.begin';
  pkg_emp.proc_log(pi_pkg => c_pkg, pi_msg => v_msg);

  BEGIN
    SELECT deptno INTO v_deptno FROM dept WHERE dname = pi_deptname;

  EXCEPTION
    WHEN OTHERS THEN
      v_deptno := NULL;
  END;

  dbms_output.put_line('Dept:' || v_deptno);

  UPDATE emp SET comm = comm - 300 WHERE deptno = v_deptno;

  v_msg := '2.update emp ';
  pkg_emp.proc_log(pi_pkg => c_pkg, pi_msg => v_msg);

  --模拟异常
  SELECT ename INTO v_ename FROM emp WHERE deptno = v_deptno;

  v_msg := '3.end ';
  pkg_emp.proc_log(pi_pkg => c_pkg, pi_msg => v_msg);

  COMMIT;

EXCEPTION
  WHEN OTHERS THEN
    v_msg := substr(SQLERRM, 1, 100);
    ROLLBACK;
```

```
    pkg_emp.proc_log(pi_pkg => c_pkg, pi_msg => v_msg);
  END proc_test_auto_transc;

END pkg_emp;
```