PLSQL

数据语言

数据语言分类

- DDL数据定义语言：create user,create table
- DCL数据控制语言：grant
- DML数据操作语言：select,insert,update,delete
- TCL事物控制语言：commit,rollback

plsql

PL/SQL是Oracle在标准SQL语言上的过程性扩展

- 允许嵌入SQL语句，定义变量和常量
- 允许过程语言结构（条件分支语句和循环语句）
- 允许使用异常来处理Oracle错误
- 可以用于创建存储过程、触发器和程序包等
- 可以用于处理业务，规则、数据库事件或给SQL命令的执行添加程序逻辑

所有的PL/SQL程序都以**块**作为基本单位

块中包含过程化语句和SQL的DML语句。这些块可以按顺序出现，也可以相互嵌套（一个块在另一个块的内部）

块的分类

- 匿名块
  - 匿名块是出现在应用程序中的没有名字且不存储到数据库中的块
  - 匿名块出现在SQL语句可以出现的地方，它们可以调用其他程序，却不能被其他程序调用
- 命名块
  - 命名块是一种带有标签的匿名块，标签为块指定了一个名称，可以在任意地方进行重复调用
  - 子程序
    - 子程序是存储在数据库中的过程（procedure）
    - 函数（function），生成之后可以被多次执行
  - 程序包
    - 程序包是存储在数据库中的一组子程序、变量定义
    - 程序包中的子程序可以被其他程序包或者子程序调用
  - 触发器
    - 触发器是一种存储在数据库中的命名块，生成之后可以被多次执行
    - 在相应的触发器事件发生之前或之后就会被执行一次或多次

```
 /*
DECLARE
定义部分：变量,常量
BEGIN
  执行部分
```

```
  EXCEPTION
    异常处理
END;
*/

-- 匿名块:TCL,DML,以分号结束
BEGIN
  -- INSERT INTO score VALUES('S008',1,100,SYSDATE);
  -- UPDATE score SET score=99 WHERE stuno ='S008';
  -- COMMIT;
  -- SELECT * FROM score WHERE stuno ='S008';不可以直接书写
END;
```

变量

```
-- 定义变量:变量名 类型(带长度);
-- 赋值: 1.:=值;2.select into
DECLARE
  names VARCHAR2(10):='匿名';-- 定义并赋值
  nos VARCHAR2(10);
BEGIN
  nos:='S001';
  nos:=&a; -- 输出窗口
  dbms_output.put_line('stuname:'||names||' nos:'||nos);
  SELECT stuname INTO names FROM student WHERE stuno=nos;
  dbms_output.put_line('stuname:'||names);

  EXCEPTION-- 异常处理
    WHEN no_data_found THEN -- 异常类型
      dbms_output.put_line('查询的学生不存在');
END;


DECLARE
  sname VARCHAR2(10);
  sno VARCHAR2(10);
BEGIN
  sno:=&nos;
  SELECT stuname INTO sname FROM student WHERE stuno=sno;
  dbms_output.put_line('stuname:'||sname);

  EXCEPTION
    WHEN no_data_found THEN
      dbms_output.put_line('查询的学生不存在');
      dbms_output.put_line('执行结束');
END;
```

常量

```
-- 常量:常量名 CONSTANT 类型:=值
-- 1.定义时赋值;2.不能重新赋值
DECLARE
   pai CONSTANT NUMBER(5,4):=3.1415;
   r NUMBER(2);
BEGIN
  r:=&r;
  dbms_output.put_line('半径为'||r||'的圆的面积为:'||pai*r*r);
END;
```

数据类型

%type

```
-- 数据类型
-- 1.%type:表名.列名%type:与表中一列的数据类型一致
DECLARE
   sphone student.phone%TYPE;
BEGIN
  SELECT phone INTO sphone FROM student WHERE stuno=&nos;
  dbms_output.put_line('sphone:'||sphone);

  EXCEPTION
    WHEN no_data_found THEN
      dbms_output.put_line('查询的学生不存在');
END;
```

%ROWTYPE

```
-- %ROWTYPE:表名%rowtype:获取表中的所有列的数据类型
-- 取值：变量.属性名
DECLARE
   stu student%ROWTYPE;
BEGIN
  SELECT * INTO stu FROM student WHERE stuno=&nos;
  dbms_output.put_line('phone:'||stu.stuname||'idcard:'||stu.idcard);

  EXCEPTION
    WHEN no_data_found THEN
      dbms_output.put_line('查询的学生不存在');
END;
```

%record

```
-- %record:自定义类型
/*
type 类型名称 is record(
   列名  类型
)
*/
DECLARE
   -- 定义类型
   TYPE stutype IS RECORD(
```

```
        sname student.stuname%TYPE,
        sidcard student.idcard%TYPE,
        states student.state%TYPE
    );
    sturow stutype;-- 定义sturow变量,变量类型是自定义的stutype类型
BEGIN
  SELECT stuname,idcard,state INTO sturow FROM student WHERE stuno=&nos;
  dbms_output.put_line('phone:'||sturow.sname||'idcard:'||sturow.sidcard);

  EXCEPTION
    WHEN no_data_found THEN
      dbms_output.put_line('查询的学生不存在');
END;
```

索引表

```
-- 索引表：表类型
/*
TYPE 类型名称 IS TABLE OF 行类型
INDEX BY 索引类型(BINARY_INTEGER/VARCHAR2(10));
索引可以不按顺序,可以是负数
赋值：变量(索引)
取值：变量(索引).属性
*/
DECLARE
   TYPE stu_table IS TABLE OF student%ROWTYPE
   -- INDEX BY BINARY_INTEGER;-10,0,10
   INDEX BY VARCHAR2(10);
   stu stu_table;
BEGIN
   SELECT * INTO stu('a') FROM student WHERE stuno=&nos1;
   -- SELECT * INTO stu('b') FROM student WHERE stuno=&nos2;
   -- SELECT * INTO stu('c') FROM student WHERE stuno=&nos3;

   dbms_output.put_line('stuname:'||stu('a').stuname);
   -- dbms_output.put_line('stuname:'||stu('b').stuname);
   -- dbms_output.put_line('stuname:'||stu('c').stuname);
END;
```

条件判断

if

```
-- 条件判断
/*
if 表达式(or/and) then
  处理
elsif 表达式 then
  处理
else
  处理
end if;
*/
DECLARE
  SCORES SCORE.SCORE%TYPE;
```

```
BEGIN
  SELECT MIN(SCORE) INTO SCORES FROM SCORE WHERE STUNO = 'S001';

  -- 0-60 不及格;60-70 及格;70-80 良好;80-90 优秀;90-100 完美
  IF SCORES < 60 AND SCORES >= 0 THEN
    DBMS_OUTPUT.PUT_LINE('不及格');
  ELSIF SCORES < 70 THEN
    DBMS_OUTPUT.PUT_LINE('及格');
  ELSIF SCORES < 80 THEN
    DBMS_OUTPUT.PUT_LINE('良好');
  ELSIF SCORES < 90 THEN
    DBMS_OUTPUT.PUT_LINE('优秀');
  ELSIF SCORES < 100 OR SCORES = 100 THEN
    DBMS_OUTPUT.PUT_LINE('完美');
  ELSE
    DBMS_OUTPUT.PUT_LINE('作弊');
  END IF;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('无查询结果');
END;
```

case

```
-- case
/*
1.
case
   when 表达式1 then 处理1
   when 表达式2 then 处理2
   ...
   else 处理
end case;
2.
case 表达式
   when 结果1 then 处理1
   when 结果2 then 处理2
   ...
   else 处理
end case;
*/
-- a.写法一
DECLARE
  SCORES SCORE.SCORE%TYPE;
BEGIN
  SELECT MIN(SCORE) INTO SCORES FROM SCORE WHERE STUNO = 'S001';

  CASE
    WHEN SCORES >= 0 AND SCORES < 60 THEN
      DBMS_OUTPUT.PUT_LINE('不及格');
    WHEN SCORES >= 60 AND SCORES <= 100 THEN
      DBMS_OUTPUT.PUT_LINE('及格');
    ELSE
      DBMS_OUTPUT.PUT_LINE('作弊');
  END CASE;
END;
```

```
-- b.写法二
DECLARE
  STATES STUDENT.STATE%TYPE;
BEGIN
  SELECT STATE INTO STATES FROM STUDENT WHERE STUNO = 'S001';

  CASE STATES
    WHEN 1 THEN
      DBMS_OUTPUT.PUT_LINE('在校');
    WHEN 0 THEN
      DBMS_OUTPUT.PUT_LINE('离校');
    ELSE
      DBMS_OUTPUT.PUT_LINE('异常');
  END CASE;
END;
```

循环

loop

```
-- 循环
/*
loop
  exit...
end loop
*/
-- exit
DECLARE
   i NUMBER(10):=1;
BEGIN
  LOOP
    dbms_output.put_line(i);
    i:=i+1;
    EXIT; -- 退出
  END LOOP;
END;

-- exit when
DECLARE
   i NUMBER(10):=1;
BEGIN
  LOOP
    dbms_output.put_line(i);
    EXIT WHEN i=10; -- 退出
    i:=i+1;
  END LOOP;
END;

-- exit+if
DECLARE
   i NUMBER(10):=1;
BEGIN
  LOOP
    dbms_output.put_line(i);
```

```
    i:=i+1;
    IF i=11 THEN
      EXIT; -- 退出
    END IF;
  END LOOP;
END;
```

while

```
-- while
/*
WHILE 循环条件 LOOP
   循环操作
   修改循环变量
END LOOP;
*/
DECLARE
   i NUMBER(10):=1;
BEGIN
  WHILE i<11 LOOP
     dbms_output.put_line(i);
     i:=i+1;
  END LOOP;
END;
```

for

```
-- for
/*
FOR 循环变量 IN [REVERSE]范围(从小到大) LOOP
     循环操作
END LOOP;
*/
DECLARE
  -- i NUMBER(10):=1;
BEGIN
  FOR i IN 1..10 LOOP
     dbms_output.put_line(i);
  END LOOP;
END;

-- 倒叙
BEGIN
  FOR i IN REVERSE 1..10 LOOP
     dbms_output.put_line(i);
  END LOOP;
END;
```

异常

```
-- 异常
DECLARE
  srow score%ROWTYPE;
  myexception EXCEPTION; -- 定义自定义异常
```

```
BEGIN
  srow.stuno:=&stuno;
  srow.cid:=&cid;
  srow.score:=&score;
  -- 判断
  IF srow.score>100 OR srow.score<-1 THEN
    -- 抛出自定义异常
    RAISE myexception;
  END IF;

  INSERT INTO score VALUES(srow.stuno,srow.cid,srow.score,SYSDATE);
  COMMIT;

  EXCEPTION -- 语法错误不会捕获
    WHEN myexception THEN
      dbms_output.put_line('成绩输入错误');
    WHEN OTHERS THEN
      dbms_output.put_line('程序错误');
END;

SELECT * FROM score WHERE stuno='S009';

-- 共享锁
SELECT course.*,ROWID FROM course;
SELECT course.* FROM course FOR UPDATE;
UPDATE course SET cname='S3' WHERE cid=3;
```

事务控制

```
-- 事务控制
BEGIN
  INSERT INTO score VALUES('S010',1,60,SYSDATE);
  INSERT INTO score VALUES('S010',2,70,SYSDATE);
  INSERT INTO score VALUES('S010',3,80,SYSDATE);
  -- COMMIT;
  ROLLBACK;
END;

-- commit,rollback;提交/回滚之前的所有事物
INSERT INTO score VALUES('S010',1,100,SYSDATE);
BEGIN
    INSERT INTO score VALUES('S010',1,60,SYSDATE);
    BEGIN
        INSERT INTO score VALUES('S010',2,70,SYSDATE);
        COMMIT;
        BEGIN
            INSERT INTO score VALUES('S010',3,80,SYSDATE);
        END;
    END;
END;
```

回滚点

```
-- savepoint
BEGIN
  INSERT INTO SCORE VALUES ('S010', 1, 60, SYSDATE);
  SAVEPOINT A;
  INSERT INTO SCORE VALUES ('S010', 2, 70, SYSDATE);
  INSERT INTO SCORE VALUES ('S010', 3, 80, SYSDATE);
  ROLLBACK TO A;
  COMMIT;
END;

SELECT * FROM score WHERE stuno='S010';
DELETE score WHERE stuno='S010';
COMMIT;
```