

第 18 章 存储过程与函数

存储过程和函数是命名的 PL/SQL 或 T-SQL 程序段，存储于数据库中，用于完成特定的数据处理任务。存储过程和函数分为系统以及用户自定义两类。

本章主要包括：

- 存储过程
- 函数
- 查询存储过程及函数的定义

18.1 存储过程

创建存储过程，Oracle 和 SQL Server 都使用 `create procedure` 语句，SQL Server 还可以把 `procedure` 关键字简写为 `proc`。

Oracle 使用 `create or replace procedure` 修改存储过程的定义，若其后的存储过程名称不存在，则创建，否则修改其定义；而 SQL Server 使用 `alter procedure` 语句修改存储过程定义。修改存储过程定义的语法与创建时相同。

18.1.1 不附带参数的存储过程

Oracle 中创建不附带参数的存储过程只要把 PL/SQL 程序段中的 `declare` 关键字替换为 `create procedure`，其他内容基本不变。

下面示例把上一章使用 `while` 循环结构求和的示例修改为存储过程：

```
SQL> create procedure sumnum
2  as
3      i int:=1;
4      mysum int:=0;
5  begin
6      while(i<=100) loop
7          mysum:=mysum+i;
8          i:=i+1;
9      end loop;
10     i:=i-1;
11     dbms_output.put_line('sum='||mysum||', '||i||);
12 end;
13 /
```

Oracle 执行存储过程有两种方式：

- 使用 `exec` 关键字
- 使用 `begin ... end` 程序段

与 SQL Server 不同，使用第一种方式时，`exec` 关键字不能省略，而使用第二种方式时，不能附加 `exec` 关键字。

使用 `exec` 关键字上述示例创建的 `sumnum` 存储过程：

```
SQL> exec sumnum
sum=5050, i=100

PL/SQL 过程已成功完成。
```

使用 begin ... end 程序段执行 sumnum 存储过程:

```
SQL> begin
  2      sumnum;
  3  end;
  4  /
sum=5050, i=100

PL/SQL 过程已成功完成。
```

在 SQL Server 中实现以上求和功能的存储过程语法如下:

```
1> create proc sumnum
2> as
3> declare @n numeric, @sum numeric
4> set @n=0
5> set @sum=0
6> while @n<101
7> begin
8>     set @sum=@sum+@n
9>     set @n=@n+1
10> end
11> print @sum
12> go
```

SQL Server 使用 exec 执行存储过程,也可以通过输入存储过程名称直接执行存储过程:

```
1> exec sumnum
2> go
5050

1> sumnum
2> go
5050
```

SQL Server 的程序段或存储过程中调用另一个存储过程时,必须使用 exec,不能省略。

18.1.2 附带输入参数的存储过程

存储过程可以附带参数,使得其功能更灵活,参数及其类型附加在存储过程名称之后,并用括号括起来。

下面示例是在 Oracle 中修改上节示例创建的 sumnum 存储过程,通过输入参数指定求和上限,并指定参数的默认值为 100:

```
SQL> create procedure sumnum2
  2  (n int:=100)
  3  as
  4      i int:=1;
  5      mysum int:=0;
  6  begin
  7      while(i<=n) loop
  8          mysum:=mysum+i;
  9          i:=i+1;
 10      end loop;
 11      i:=i-1;
 12      dbms_output.put_line('sum='||mysum||', '||i||');
 13  end;
 14  /
```

执行上述示例,指定参数为 100,求出前 100 个自然数之和:

```
SQL> exec sumnum2(100);  
sum=5050, i=100
```

PL/SQL 过程已成功完成。

或不指定参数值使其取默认值 100:

```
SQL> exec sumnum2;  
sum=5050, i=100
```

PL/SQL 过程已成功完成。

SQL Server 创建附带输入参数的存储过程时, 直接把参数名称及类型附加在存储过程名称之后, 不需要用括号把参数括住。

创建与上面 Oracle 示例相同功能的存储过程, SQL Server 的语法如下:

```
1> create proc sumnum2  
2> @max numeric=100  
3> as  
4> declare @n numeric, @sum numeric  
5> set @n=0  
6> set @sum=0  
7> while @n<@max+1  
8> begin  
9>     set @sum=@sum+@n  
10>    set @n=@n+1  
11> end  
12> print @sum  
13> go
```

指定输入参数为 10, 执行上述存储过程求出前 10 个自然数之和:

```
1> sumnum2 10  
2> go  
55
```

不指定参数, 则参数取默认值 100:

```
1> exec sumnum2  
2> go  
5050
```

下面示例创建存储过程, 查询 emp 表中指定 empno 的记录的 ename 及 sal 值:

```
1> create proc searchbyeno  
2> @eno numeric(4)  
3> as  
4> select ename,sal from emp where empno=@eno  
5> go
```

执行上述存储过程, 查询 empno 为 7369 的记录:

```
1> searchbyeno 7369  
2> go  
ename      sal  
-----  
SMITH      2900.00
```

18.1.3 附带输出参数的存储过程

输出参数类似于函数中的返回值，在 Oracle 数据库中创建存储过程时，使用 out 关键字来标识输出参数。

在执行附带输出参数的存储过程时，要先定义接收输出参数值的变量，并按照参数的顺序把这个变量传递给存储过程，执行完毕后，在程序段中即可引用输出参数的值。

下面示例使用输出参数得到两个整数的乘积：

```
SQL> create procedure mul
  2  (n1 int, n2 int, m out int)
  3  as
  4  begin
  5      m:=n1*n2;
  6  end;
  7  /
```

执行时，定义变量 a 用于接收存储过程的输出参数，并在调用存储过程时，按照顺序传入输出参数 a，传入时不需要指定 out 关键字：

```
SQL> declare a int;
  2  begin
  3      mul(2,3,a);
  4      dbms_output.put_line(a);
  5  end;
  6  /
```

PL/SQL 过程已成功完成。

SQL Server 存储过程的输出参数用 output 标识，与 Oracle 的 out 功能相同，在执行存储过程时，也要预先定义相应变量接收输出参数的值。

下面示例创建存储过程，使用输出参数的方法，返回两个整数的乘积：

```
1> create proc mulnum
2> @n1 int,
3> @n2 int,
4> @result int OUTPUT
5> as
6> set @result=@n1*@n2
7> go
```

执行上述存储过程时，要先定义变量用于接收输出参数的值，执行时，把此变量按照创建存储过程时的顺序传递给存储过程，与 Oracle 不需要附加 out 关键字不同，SQL Server 要求附加 output 标识：

```
1> declare @r int
2> exec mulnum 12,5,@r output
3> print @r
4> go
60
```

注意：这里的 exec 不能省略。

18.2 函数

函数与存储过程的区别主要是有返回值，在函数定义头部，除了要定义输入参数以外，

还要说明返回值的数据类型。使用函数时，可以直接引用函数的执行结果，而不必预先定义变量来接收。

Oracle 和 SQL Server 都使用 `create function` 语句创建函数，如果要修改函数定义，Oracle 使用附带 `or replace` 的 `create function` 语句，即 `create or replace function`，若其后的函数名称不存在，则创建，否则修改其定义；SQL Server 使用 `alter function` 语句修改函数定义。

18.2.1 Oracle 的函数

下面示例创建的函数用于返回两个整数的乘积：

```
SQL> create function mul2(n1 int, n2 int)
  2  return int
  3  as
  4  m int;
  5  begin
  6      m:=n1*n2;
  7      return m;
  8  end;
  9  /
```

在 `select` 语句中调用上述函数：

```
SQL> select mul2(3,4) from dual;

MUL2(3,4)
-----
        12
```

创建函数时，也可以直接返回乘积，而不使用第三个变量：

```
SQL> create or replace function mul2(n1 int, n2 int)
  2  return int
  3  as
  4  begin
  5      return n1*n2;
  6  end;
  7  /
```

18.2.2 SQL Server 的函数

T-SQL 创建函数时，`return` 关键字要使用第三人称，附加 `s`，另外函数主体中的变量定义要包含在 `begin ... end` 程序块内，这与 PL/SQL 不同，语法的其他方面都是相同的。

下面示例创建的函数用于返回两个整数的乘积：

```
1> create function mul(@n1 as int, @n2 as int)
2> returns int
3> as
4> begin
5>     declare @m as int
6>     set @m=@n1*@n2
7>     return @m
8> end
9> go
```

执行上述函数：

```
1> select dbo.mul(2,3)
2> go
```

也可以直接返回乘积:

```

1> alter function mul(@n1 as int, @n2 as int)
2> returns int
3> as
4> begin
5>     declare @m as int
6>     return @n1*@n2
7> end
8> go
1> select dbo.mul (3, 4)
2> go

```

18.3 查询存储过程及函数的定义

Oracle 的静态数据字典视图 dba_source 可以用于查询所有可编程对象的定义文本。
如查询存储过程 sumnum 的定义:

```

SQL> select text from dba_source
      2  where name='SUMNUM'
      3  and owner='SCOTT'
      4  /

```

查询函数 mul2 的定义:

```

SQL> select text from dba_source
      2  where name='MUL2'
      3  and owner='SCOTT'
      4  /

```

SQL Server 查询存储过程或函数的定义文本, 可以使用下几种方法:

- 使用 sp_helptext 系统存储过程
- 使用 sys.sql_modules 目录视图
- 使用 INFORMATION_SCHEMA.ROUTINES 视图

下面只说明查询存储过程 sumnum 定义的方法, 查询函数定义, 只需要把查询条件中的存储过程名称替换为函数名称。

使用 sp_helptext:

```

C:\>sqlcmd -E -d law -Y 40 -y 20
1> sp_helptext sumnum
2> go

```

使用数据字典视图 INFORMATION_SCHEMA.ROUTINES:

```

C:\>sqlcmd -E -d hr -Y 500 -y 50
1> select routine_definition from INFORMATION_SCHEMA.ROUTINES
2> where routine_name='sumnum'
3> go

```

使用 sys.sql_modules 目录视图:

```
C:\>sqlcmd -d law -y 500
1> select definition
2> from sys.sql_modules
3> where object_name(object_id)='sumnum'
4> go
```