Chapter 5

第 5 章　SQL 语言高级功能

SQL 语言除了最基本的数据定义功能和数据操纵功能外，还具备数据控制、复杂流程处理等高级功能。本章重点介绍 SQL 的数据控制语言、控制复杂流程的存储过程和触发器，以及嵌入式 SQL。

5.1　数据控制语言

数据控制语言是用来设置或者更改数据库用户或角色权限的语句，这些语句包括 GRANT、REVOKE 等，在默认状态下，只有 sysadmin、dbcreator、db_owner 或 db_securityadmin 等角色的成员才有权执行数据控制语言。

5.1.1　权限和角色

数据库中的权限分为两类：一类是维护数据库管理系统的权限，另一类是操作数据库中的对象和数据的权限。后者又包括两种：一种是操作数据库对象的权限，如创建、修改和删除数据库对象；另一种是操作数据库中数据的权限，包括对表或视图中的数据进行增、删、改和查询操作。对不同的数据库对象有不同的操作权限，标准 SQL 也通过简单的单词来表达这些权限（见表 5-1）。

表 5-1　不同对象类型允许的权限

对象类型	权　限	含　义
TABLE	SELECT	查询表的全部属性列的权限
	UPDATE	更新表中数据的权限
	INSERT	向表中插入数据的权限
	DELETE	从表中删除数据的权限
	REFERENCES	引用表的权限
	ALTER	修改表结构的权限
	INDEX	在表上建索引的权限
	ALL PRIVILEGES	所有权限

续表

对象类型	权　限	含　义
TABLE	SELECT(列名)	查询表的指定属性列的权限
	UPDATE(列名)	更新表中指定属性列的数据权限
	INSERT(列名)	向表中指定属性列插入数据的权限
	DELETE(列名)	从表中指定属性列删除数据的权限
DATABASE	CREATE(或 ALTER) TABLE	在数据库中创建(或修改)表的权限
	CREATE(或 ALTER) VIEW	在数据库中创建(或修改)视图的权限
	CREATE(或 ALTER) FUNCTION	在数据库中创建(或修改)函数的权限
	CREATE(或 ALTER) PROCEDURE	在数据库中创建(或修改)存储过程的权限
	CREATE(或 ALTER) TRIGGER	在数据库中创建(或修改)触发器的权限
	BACKUP DATABASE	备份数据库的权限
	BACKUP LOG	备份日志的权限
	CONNECT	连接数据库的权限

其中,在数据库创建表的权限属于 DBA,DBA 可以将这个权限授给普通用户,普通用户拥有此权限可以建立基本表,并成为该表的属主(Owner)。一旦成为一个表的属主,将拥有对该表的一切操作权限。

在数据库中,为了便于管理用户及其权限,借用了现实生活中角色的概念。角色是一组权限的集合。在角色里,可以把一些基本权限进行组合,然后把角色整体授予一个用户或一组用户,这样可以降低数据库管理员的工作负担和维护成本。例如,在超市数据库管理系统中,假设采购员的权限有查看供应商、查看商品库存信息、修改商品价格等 10 个权限。如果超市里有 10 位采购员,那么数据库管理员将分别对 10 位采购员用户依次赋予 10 个权限,共需要进行 10×10=100 次的授权操作。如果把这 10 个权限组合绑定,赋予一个角色,暂且定名为“采购员”,那么数据库管理员对 10 个采购员的授权工作将由 100 次缩减为 10 次,只要被授予“采购员”角色的用户在使用数据库的过程中就扮演了“采购员”一职,一旦其调换岗位或因其他原因不从事采购工作,数据库管理员直接把角色“采购员”的权限收回即可。DBA 可以把企业中类似的相对固定的岗位权限都采用角色的方式来定义,从而大大减轻其工作负担。

5.1.2 授权语句

授权语句实现对数据库中数据的存取控制。存取控制是指授予某个用户某种特权,利用该特权能够以某种方式(如读取、修改等)访问数据库中的某些数据对象。

大型数据库管理系统几乎都支持自主存取控制,目前的 SQL 标准也对自主存取控制提

供支持,主要通过 SQL 的 GRANT 语句和 REVOKE 语句来实现。GRANT 将某种权限授予某个用户,GRANT 语句的一般格式为:

```
GRANT<权限>[,<权限>]...
[ON <对象类型><对象名>]
TO<用户>[,<用户>]...
[WITH GRANT OPTION];
```

其语义为:将对指定数据对象的指定操作权限授予指定的用户。接受权限的用户可以是一个或多个具体用户;也可以是 PUBLIC,即全体用户。

如果指定了 WITH GRANT OPTION 子句,则获得某种权限的用户还可以把这种权限再授予其他用户。如果没有指定 WITH GRANT OPTION 子句,则获得某种权限的用户只能使用该权限,但不能传播该权限。

假设 DBA 已经通过 DBMS 提供的工具建立了以下用户:user1、user2、user3、user4、user5。下面是利用 GRANT 语句进行授权操作的一些例子。

【例 5-1】 DBA 把在数据库 TEST 中的建表的权限授予用户 user1。

标准 SQL 语句的写法:

```
GRANT CREATETAB ON DATEBASE TEST TO user1;
```

因具体的数据库管理系统设计的不同,实现这个命令的方式也略有差异,如在 SQL Server 中使用的命令就是:

```
GRANT CREATE TABLE TO user1;
```

一旦授权成功,user1 拥有了在 TEST 中创建表的权限。

【例 5-2】 把对基本表 Goods 的查询权限授予所有用户。

```
GRANT SELECT ON Goods TO PUBLIC;
```

关键字 PUBLIC 是个系统预先建立好的角色。命令一旦执行成功,则数据库中的所有拥有 PUBLIC 角色的用户都拥有了对 Goods 表的查询权限。

【例 5-3】 把对基本表 Supplier 的查询权限授予用户 user2。

```
GRANT SELECT ON Supplier TO user2;
```

【例 5-4】 把对基本表 Supplier 的所有权限授予用户 user3。

```
GRANT ALL PRIVILEGES ON Supplier TO user3;
```

【例 5-5】 把对基本表 Goods 的查询权限和修改价格的权限授予用户 user4。

```
GRANT SELECT,UPDATE(Price) ON Goods TO user4;
```

【例 5-6】 把对基本表 Supply 的插入权限授予用户 user5,并允许他将此权限授予其他用户。

```
GRANT INSERT ON Supply TO user5 WITH GRANT OPTION;
```

执行 SQL 语句后，user5 不仅拥有对表 Supply 的 INSERT 权限，还可以传播此权限，即通过授权命令把这个权限授给其他用户，如 user5 可以执行以下命令将该权限授予 user4 和 user3：

```
GRANT INSERT ON Supply TO user4, user3;
```

因授权命令中没有 WITH GRANT OPTION，则 user4 不能再传播此权限。

从上述例子可以看出，授权语句不仅可以一次向一个或多个用户授权，而且还可以一次传播一个或多个同类对象的多个权限。

5.1.3 收回权限

授予的权限可以由 DBA 或其他授权者用 REVOKE 语句收回，REVOKE 语句的一般格式为：

```
REVOKE <权限>[,<权限>]...
    [ ON <对象类型><对象名>]
    FROM <用户>[,<用户>]... ;
```

【例 5-7】 把用户 user4 对基本表 Goods 的修改价格的权限收回。

```
REVOKE UPDATE (Price) ON TABLE Goods FROM user4;
```

收回权限的具体细节与授权类似，这里不再一一举例说明。读者可以参照授权的例子，自己写出收回这些权限的语句。

5.2 存储过程

SQL 是一种非过程结构化查询语言，有很强的表达能力。基于数据库的应用逻辑日益复杂，在应用中仅仅使用顺序执行的 SQL 语句序列表现出很大的局限性，减少局限性的方法之一是使用 PL/SQL 来编写存储过程。PL/SQL 是编写数据库应用模块的一种过程语言，它结合了 SQL 的数据操作能力和过程化语言的流程控制能力，开发人员可以设计出更加灵活的应用系统。

存储过程(Stored Procedure，SP)是使用 PL/SQL 编写的模块，存储过程经编译和优化后存储在数据库服务器端的数据库中，使用时调用即可。与表、视图一样，存储过程也是一直存储在数据库中的对象。存储过程与其他编程语言中的过程类似，可以接收输入参数并以输出参数的形式向调用过程返回多个值。存储过程与函数不同，因为存储过程不返回取代其名称的值，因此不能直接在表达式中使用。

存储过程可以使用户用 SQL 语句和流程控制语句对数据进行方便的处理，增强了

SQL 的功能和灵活性，完成 SQL 很难完成的复杂的逻辑判断和比较复杂的运算；存储过程还可以优化性能，在运行存储过程前，数据库服务器对其进行语法和语义分析，然后进行预编译，使用时直接调用即可，执行速度较高；另外，由于存储过程保存在服务器端，可以降低网络的通信量，同时一旦用户规则发生变化，只在服务器端修改存储过程、重新发布相关的数据库数据就可以了。总之，方便部署应用。

存储过程的功能很多。例如，可以把完成某一数据库处理的功能设计为一个存储过程，就可以被各个程序反复调用，从而减轻程序编写的工作量。因此，在数据库应用系统中，充分利用存储过程来完成应用系统的逻辑操作处理，可以提高系统的运行性能和可维护性。

5.2.1　创建存储过程

创建存储过程的 SQL 语句语法格式为：

```
CREATE PROCEDURE <存储过程名>([@参数 1 数据类型 IN/OUT/ INOUT,@参数 2 数据类型
                              IN/OUT/ INOUT...])AS
[DECLARE <内部变量>]
BEGIN
    <SQL 语句>
END;
```

其中，@参数列表指明参数名称和数据类型，可以带多个参数，参数可以是输入参数、输出参数或输入输出参数，其中，输入参数用 IN 标记，默认状态下都是输入参数；输出参数用 OUT 特别说明；输入输出参数用 INOUT 标记。DECLARE 可选项定义存储过程内部的变量，不需要变量时，可以无此项。在 BEGIN…END 之间定义存储过程要执行的 SQL 语句，可以是一组 SQL 语句，也可以包含流程控制语句等。

【例 5-8】　创建一个存储过程。

```
CREATE PROCEDURE sp_getname
As
Select Sname FROM Supplier;
```

这是一个名为 sp_getname 的存储过程，它在第一次执行时存放在数据库中，当以后需要从供应商中提取供应商名称时，直接调用这个存储过程即可，DBMS 在服务器端完成查询，并将结果传送给客户。

【例 5-9】　创建一个带 IN 参数的存储过程。

```
CREATE PROCEDURE sp_getname_D (@addr char(20) IN)
As
Select Sname FROM Supplier WHERE Saddress=@addr;
```

这个存储过程只选择给定地区的供应商名称，指定地区的参数通过@addr 传入。

【例 5-10】　创建一个带 OUT 参数的存储过程。

```
CREATE PROCEDURE sp_getSUM (@TOTAL FLOAT OUT)
As
Select @TOTAL=SUM(Price) FROM Goods;
```

这个存储过程计算所有商品的总市值，并把计算的结果输出到 OUT 参数@TOTAL。程序调用存储过程 sp_getSUM 后，可以直接将其赋值给一个变量。

5.2.2 修改存储过程

SQL 中修改存储过程可以用 ALTER PROCEDURE 命令，其语法格式为：

```
ALTER PROCEDURE <存储过程名>([@参数 1 数据类型 IN/OUT/ INOUT,@参数 2 数据类型
                              IN/OUT/ INOUT...])AS
[DECLARE
<内部变量>]
BEGIN
    <SQL 语句>
END;
```

各参数的含义与创建存储过程的 CREATE PROCEDURE 语句中的参数相同。

【例 5-11】 修改例 5-10 存储过程，使得存储过程返回商品的平均价格。

```
ALTER PROCEDURE sp_getSUM (@TOTAL FLOAT OUT)
As
Select @TOTAL=AVE(Price) FROM Goods;
```

5.2.3 删除存储过程

不再需要的存储过程可以使用删除命令删除，语法格式为：

```
DROP PROCEDURE <存储过程名>;
```

【例 5-12】 删除存储过程 sp_getname_D。

```
DROP PROCEDURE sp_getname_D;
```

5.2.4 执行存储过程

在 SQL 中，通过 Execute(简写 exec)命令执行已经创建成功的存储过程。Execute 命令的语法格式如下：

```
Exec|execute
{
```

```
[@return_status=]
{module_name[;name]|@module_name_var}
[[@parameter=]{value|@variable[output]|default]}]
[,...n]
[with recompile]
}
[;]
```

参数说明见表 5-2。

表 5-2　参数说明

参　数　名	说　　明
@return_status	可选的整型变量，代表存储过程的返回状态，该变量在用于 Execute 语句前，必须已经声明过
module_name	要调用的存储过程名，必须符合标识符命名规则
[name]	可选整数，用于对同名的过程分组
@module_name_var	代表存储过程名单局部变量
@parameter	存储过程的参数，必须和存储过程定义中的相同
Value	传递的参数值。如果参数名称没有指定，参数值的顺序必须和存储过程中定义的顺序相同
@variable	存储参数或返回参数的变量
output	指定该参数为输出参数，该参数在存储过程定义时必须已经使用 output 声明为输出参数。必须使用变量来接收输出值，而且该变量必须事先声明
default	指明该参数使用默认值。如果某参数定义时没有指定默认值，则不能使用 default 关键字
with recompile	一般情况下，存储过程只有在第一次执行时，系统对其进行编译，并将其存储起来，以后执行时直接取出执行计划执行，不再编译。使用 with recompile，强制在执行存储过程时重新对其进行编译

【例 5-13】 执行存储过程 price_query。

```
DECLARE @t float
EXECUTE price_query  '002',@t output
SELECT @t;
```

注意：如果存储过程含义参数并且没有指定默认值，则调用存储过程时必须对参数赋值，可以使用两种方式传递参数：按顺序赋值和按名赋值。

1. 按顺序赋值

按属性给参数赋值时，不需要给出参数的名称，并且调用语句中值的顺序必须和存储过程定义中参数定义的顺序保持一致。如果某参数有默认值，可以使用 default 指明该参数使

用默认值，如果该参数位于参数列表的末尾，则 default 可以省略。如例 5-13 即是按顺序赋值的方式。

2. 按名赋值

采用“参数名＝参数值”的方式对参数进行赋值，赋值的顺序跟存储过程定义时的顺序可以不一致。如例 5-13 可以改为：

```
DECLARE @t float
EXECUTE price_query @gno='002',@price=@t output
SELECT @t;
```

5.2.5 过程声明

过程声明 DECLARE 定义存储过程 SQL 语句中使用的所有局部变量或常量，在执行存储过程时，DBMS 为它们分配空间，并可以在过程体内被引用。变量声明时系统进行如下的处理过程：

(1) 指定局部变量名称，名称的第一个字符必须是@。

(2) 指定变量的数据类型，可以是系统提供的数据类型或用户自定义的数据类型，对于有长度设置的还可以设置长度，如字符型数据可以设置字符的长度，数值型可以设置数值的精度和小数位数。

(3) 为变量赋初值。

声明语句的语法为：

```
DECLARE <变量名称 数据类型[默认值]>[,...]
```

可以一次定义多个变量，各变量之间用逗号隔开。

【例 5-14】 定义一个变量，记录平均价格。

```
DECLARE
@avgprice FLOAT;
```

另外，过程声明中定义的变量允许设定默认值或使用约束。

【例 5-15】 定义商品起始价格，默认为 0。

```
DECLARE
@gprice FLOAT 0;
```

存储过程中可用的数据类型一般和数据库服务器所支持的内部数据类型兼容，同时多数数据库服务器还可能支持记录类型或列类型，在 SQL Server 中有 table 数据类型。表声明包括列定义、名称、数据类型和约束。允许的约束类型只包括 PRIMARY KEY、UNIQUE、NULL 和 CHECK。这两种方式的基准记录或列来自数据库中已经存在的某个表的行定义或列，变量的类型和数据库对象的类型保持同步，保证使用这些变量输入输出相关对象时的类型完全一致。

【例 5-16】 定义商品变量。

```
Decalre @goodrow TABLE (Gno CHAR(20) PRIMARY KEY,
                        Gname CHAR(50) NOT NULL,
                        Price FLOAT,
                        BATCH CHAR(20));
```

这里定义的@goodrow 变量与 TABLE 表定义的结构相同。

注意：过程声明还有更多内容，许多内容依赖于具体的数据库服务器，包括定义的语法和后面几个小节中的内容，但总体上符合或基本遵循新的 SQL 标准所规定的用法。另外，还要注意在某些数据库服务器中，过程体中不能定义和数据库对象同名的过程变量。

5.2.6 基本语句和表达式

存储过程一般用来完成数据查询和数据处理操作，这些操作可以用执行语句来完成。在执行语句中主要包括赋值语句、查询语句和过程/函数调用语句。

1. 赋值语句

存储过程在过程声明 DECLARE 中定义了过程变量后，在过程体中就可以对变量赋值了，常用的方式使用 SET 来修改过程定义的局部变量或过程参数的值。赋值变量的值可以是常量、变量、函数和表达式，还可以是子查询。

【例 5-17】 将例 5-14 中定义的平均价格变量提高 1%。

```
SET @avgprice=@avgprice * 1.01;
```

【例 5-18】 将 Goods 表的平均价格输出到例 5-14 中定义的平均价格变量中。

```
SET @avgprice=(SELECT AVG(Price) FROM Goods);
```

2. 查询语句

存储过程可以包含很多的 SQL 语句，这些 SQL 语句都是顺序执行的单语句，语句中可以使用过程所声明的变量。

【例 5-19】 统计商品的平均价格。

```
SELECT AVG(Price) INTO @avgprice FROM Goods;
```

本例中，查询语句的输出 AVG(PRICE)用 INTO 子句将输出保存在过程变量 avgprice 中。

3. 过程/函数调用语句

在 PL/SQL 中，数据库服务器一般支持在过程体中调用其他存储过程或函数。可以使用 PERFORM、CALL 或直接使用 SELECT 等多种不同方式激活对其他过程/函数的执行。

在存储过程内部发出过程/函数调用时可以使用过程声明的变量或接口参数作为函数调用的参数。如果被调用函数含有输出或输入输出(OUT/INOUT)参数，则调用时传入的变量在被调用过程中发生的修改在当前过程中可见。

4. 表达式

在存储过程体中，一般都会使用常量、变量、操作符、标识符组合成各种表达式，并在运算后得到一个合法的结果。

常量是表示一个特定数据值的符号，在程序运行过程中其值保持不变，常量的格式取决于它所表达的值的数据类型，如字符串常量、数值常量、日期时间常量和空值。

变量是可以对其赋值并参与运算的一个实体，其值在运算过程中可以发生改变，变量可以分为全局变量和局部变量两类，其中全局变量由系统定义并维护，局部变量由用户定义并赋值。

存储过程中可使用的操作符包括算术运算符、赋值运算符、字符串串联运算符、比较运算符、逻辑运算符、按位运算符和一元运算符，各种运算符的优先级见表 5-3，其中比较运算和逻辑运算都属于布尔表达式，结果为逻辑值真或假。在流程控制的各种控制语句中，控制条件一般使用布尔表达式。

表 5-3　运算符优先级

优先级	运算符名称	符　号	语　　法	含　　义
最高	按位取反	～	～表达式	一元运算符：只对一个表达式按位取反
	乘	*	表达式 1 * 表达式 2	算术运算符：对两个表达式执行乘运算
	除	/	表达式 1/表达式 2	算术运算符：对两个表达式执行除运算
	取余	%	表达式 1%表达式 2	算术运算符：对两个表达式执行取余运算
	正	＋	＋数值表达式	一元运算符：对一个表达式取正
	负	－	－数值表达式	一元运算符：对一个表达式取负
	加	＋	字符串 1＋字符串 2	字符串串联运算符：将两个字符串连接起来
	减	－	表达式 1－表达式 2	算术运算符：对两个表达式执行减运算
	按位与	&	表达式 1& 表达式 2	按位运算符：按位与
	按位异或	^	表达式 1^表达式 2	按位运算符：按位异或
	按位或	|	表达式 1|表达式 2	按位运算符：按位或
	等于	＝	表达式 1＝表达式 2	比较运算符：两个表达式是否相等
	大于	＞	表达式 1＞表达式 2	比较运算符：表达式 1 是否大于表达式 2
	大于等于	＞＝	表达式 1＞＝表达式 2	比较运算符：表达式 1 是否大于等于表达式 2
	小于	＜	表达式 1＜表达式 2	比较运算符：表达式 1 是否小于表达式 2
	小于等于	＜＝	表达式 1＜＝表达式 2	比较运算符：表达式 1 是否小于等于表达式 2
	不等于	＜＞或!＝	表达式 1＜＞表达式 2	比较运算符：两个表达式是否不等
	不大于	!＞	表达式 1!＞表达式 2	比较运算符：表达式 1 是否不大于表达式 2
	不小于	!＜	表达式 1!＜表达式 2	比较运算符：表达式 1 是否不小于表达式 2
最低	取反	Not	Not 表达式	逻辑运算符：对表达式的值取反