数据库

SHUJUKU

202X

微信公众号：灰灰考研

第六章 存储过程与事务

- 掌握变量的定义
- 掌握基本的流程控制语句
- 了解存储过程和存储函数
- 了解存储程序的类型
- 理解存储过程的作用
- 掌握存储过程的创建、修改和执行
- 掌握存储函数的创建和执行
- 掌握事务处理的过程
- 理解事务的隔离级别

一、MySQL程序设计基础

(1)变量

在MySQL中最常见的变量类型有局部变量和用户自定义变量两种在MySQL中最常见的变量类型有局部变量和用户自定义变量两种。

1. 局部变量局部变量一般定义在SQL的语句块中，常用于存储过程和存储函数的BEGIN/END语句块。局部变量的作用域只限于定义它的语句块，语句块执行完毕后，局部变量也随之释放。要定义局部变量必须使用DECLARE语句来声明，定义的同时可以使用DEFAULT子句对局部变量进行初始化赋值。DECLARE语句格式如下：

```
DECLARE var_name[,…] type [DEFAULT value];
```

value是给局部变量提供的一个默认值，包含在DEFAULT子句中。如果没有DEFAULT子句，局部变量的初始值为NULL。例如：

```
DECLARE num  int  DEFAULT 0;
```

上述例子定义一个整型局部变量num并设置其初始值为0。

```
SET var_name = expr [, var_name = expr] ;
```

```
SET num=10;
```

☆ SET语句既可以用于局部变量的赋值，也可以用于用户自定义变量的声明及赋值。

2. 用户自定义变量用户自定义变量的名字以“@”开头，形如：@var_name。为了在不同SQL语句中进行值的传递，可以把一些数值存储在用户自定义变量中，不同的SQL语句都可以访问。用户自定义变量在客户端和数据库的连接建立后被定义，直到连接断开时，用户变量才会被释放。用户自定义变量无需用DECLARE关键字进行定义，可以直接使用。例如：

```
SET @c1=1, @c2=2, @c3=4;
```

查看用户变量的值可以使用SELECT语句，例如：

```
SELECT @c1, @c2, @c3;
```

查询结果：

```
mysql> SET @c1=1, @c2=2, @c3=4;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @c1, @c2, @c3;
+------+------+------+
| @c1  | @c2  | @c3  |
+------+------+------+
|    1 |    2 |    4 |
+------+------+------+
1 row in set (0.00 sec)
```

（2）流程控制语句

在编写存储过程和存储函数时，可以使用流程控制语句对SQL语句进行组织，使其成为符合业务逻辑的代码块。MySQL中常见的流程控制语句主要有：IF语句、CASE语句、LOOP语句、WHILE语句、ITERATE语句、REPEAT语句等。1. IF语句IF语句根据逻辑判断条件的值是TRUE还是FALSE，转去执行相应的分支中的语句。IF语句的语法格式如下：

```
IF expr_condition THEN statement_list
[ELSEIF expr_condition THEN statement_list]
[ELSE statement_list]
END IF;
```

IF语句用于实现分支判断的程序结构。在上述语法格式中，expr_condition代表逻辑判断条件，statement_list代表一条或多条SQL语句。如果expr_condition的值为TRUE，则执行THEN后面的SQL语句块；如果expr_condition的值为FALSE，则执行ELSE后面的SQL语句块。下面举一个IF语句的例子，代码如下：

```
IF x>y
THEN SELECT  x;
ELSE SELECT  y;
END IF;
```

2. CASE语句CASE也是一个条件判断语句，用于多分支判断的程序结构，常用语法格式如下：

```
CASE case_expr
  WHEN when_value THEN statement_list
  [WHEN when_value THEN statement_list]……
  [ELSE statement_list]
END CASE;
```

其中，case_expr是一个表达式，when_value表示case_expr表达式可能的匹配值。如果某一个when_value的值与case_expr表达式的值相匹配，则执行对应THEN关键字后的statement_list中的语句。如果所有when_value的值与case_expr表达式的值都不匹配，则执行ELSE关键字后的statement_list中的语句。下面给出一个CASE的代码片段，根据人名显示这个人的特征：

```
CASE name
  WHEN 'sam' THEN SELECT 'young';
  WHEN 'lee' THEN SELECT 'handsome' ;
  ELSE SELECT 'good';
END CASE;
```

3. LOOP语句LOOP是一个循环语句，用来实现对一个语句块的循环执行。LOOP语句并不能进行条件判断来决定何时退出循环，会一直执行循环语句。如果要退出循环，需要使用LEAVE等语句。LOOP语句的语法格式如下：

```
[loop_label:] LOOP
        Statement_list
    END LOOP [loop_label];
```

小练习：使用LOOP语句计算1+2+3…+100。

4. LEAVE语句LEAVE语句基本语法结构如下：

```
LEAVE label;
```

5. ITERATE语句与LEAVE语句结束整个循环的功能不同，ITERATE语句的功能是结束本次循环，转到循环开始语句，进行下一次循环。ITERATE语句的格式如下：

```
ITERATE lable
```

小练习：使用ITERATE语句跳转显示1到5，然后使用LEAVE语句退出整个循环。

6. REPEAT语句REPEAT语句用于循环执行一个语句块，执行的流程是先执行一次循环语句块再进行条件表达式判断，如果条件表达式值为TRUE，则循环结束，否则再重复执行一次循环语句块。REPEAT语句的格式如下：

```
[repeat_lable:] REPEAT
        statement_list
        UNTIL expr_condition
        END REPEAT [repeat_lable];
```

其中，repeat_lable为循环标记名称，是可选的。statement_list中的语句将会被循环执行，直到expr_condition表达式值为TRUE时才结束循环。

小练习：使用REPEAT语句计算1+2+3+…+100。

7. WHILE语句WHILE语句也用于循环执行一个语句块，但是与REPEAT语句不同，WHILE语句在执行时会首先判断条件表达式是否为TRUE，如果为TRUE则继续执行一次循环语句块，执行完后再判断条件表达式。如果条件表达式的值为FALSE，则直接退出循环。WHILE语句的格式如下：

```
[while_lable:] WHILE expr_condition DO
        Statement_list
    END WHILE [while_lable];
```

小练习：使用WHILE语句计算1+2+3+…+100。

（3）光标

使用SQL语句对表中数据进行查询时，可能返回很多条记录。如果需要对查询结果集中的多条记录进行逐条读取，则需要使用光标。

1. 光标的声明要使用光标对查询结果集中的数据进行处理，首先需要声明光标。光标的声明必须在声明变量和条件之后，声明处理程序之前。光标的声明格式如下：

```
DECLARE cursor_name CURSOR FOR select_statement;
```

其中，cursor_name表示光标的名字，select_statement是光标的SELECT语句，返回一个用于创建光标的查询结果集。下面的代码声明一个名为cur_teacher的光标：

```
DECLARE cur_teacher CURSOR FOR SELECT name, age
FROM teacher ;
```

2. 光标的使用光标在使用之前必须先打开。MySQL中使用OPEN关键字打开光标。从光标查询结果集中取出一条记录可用FETCH语句。其语法格式如下：

```
OPEN cursor_name ;
FETCH cursor_name INTO var_name[,var_name…] ;
```

代码使用名为cur_teacher的光标，将查询结果集中一条记录的name和age字段的值存入teacher_name和teacher_age变量中。

```
FETCH cur_teacher INTO teacher_name,teacher_age ;
```

注意！teacher_name和teacher_age变量必须在声明光标之前定义。

3. 光标的关闭MySQL中使用CLOSE关键字来关闭光标。其语法格式如下：

```
CLOSE cursor_name ;
```

光标关闭之后就不能再使用FETCH语句从光标查询结果集中取出数据了。每执行一次FETCH语句从查询结果集中取出一条记录，将该记录字段的值送入指定的变量，通过循环可以逐条访问查询结果集中的所有记录。

以下代码使用光标查询并显示teacher表中所有教师的姓名tname和学历tedu字段的信息。

```
    DECLARE  no_more_record INT DEFAULT 0;
    DECLARE  t_name varchar(20);
    DECLARE  t_edu  varchar(20);
    DECLARE  cur_record CURSOR FOR
    SELECT  tname, tedu FROM teacher;  / *对光标进行定义* /
    DECLARE  CONTINUE HANDLER FOR NOT FOUND
    SET  no_more_record = 1;/ *错误处理，针对 NOT FOUND错误，当没有
记录时将no_more_record
赋值为1* /
     OPEN  cur_record; / *使用OPEN打开光标* /
     WHILE no_more_record != 1 DO
        FETCH  cur_record INTO t_name, t_edu;
        SELECT t_name, t_edu;
     END WHILE;
    CLOSE  cur_record;  / *用CLOSE语句把光标关闭* /
```

二、存储过程概述

存储过程是数据库服务器上一组预先编译好的SQL语句的集合，作为一个对象存储在数据库中，可以被应用程序作为一个整体来调用。在调用过程中，存储过程可以从调用者那里接收输入参数，执行后再通过输出参数向调用者返回处理结果。

（1）存储过程的基本概念

在进行数据库开发的过程中，数据库开发人员经常把一些需要反复执行的代码放在一个独立的语句块中。这些能实现一定具体功能、独立放置的语句块，我们称之为“过程”（Procedure）。

MySQL的存储过程（Stored Procedure），就是为了完成某一特定功能，把一组SQL语句集合经过编译后作为一个整体存储在数据库中。用户需要的时候，可以通过存储过程名来调用存储过程。

（2）存储程序的类型

在MySQL中，存储程序的方式主要分为以下四种。

① 存储函数（stored function）。根据调用者提供的参数进行处理，最终返回调用者一个值作为函数处理结果。

② 存储过程（stored procedure）。一般用来完成运算，并不返回结果。需要的时候可以把处理结果以输出参数的形式传递给调用者。

③ 触发器（trigger）。当执行INSERT、UPDATE、DELETE等操作时，将会引发与之关联的触发器自动执行。

④ 事件（event）。事件是根据时间调度器在预订时间自动执行的存储程序。

（3）存储过程的作用

MySQL存储过程具有以下作用。

① 存储过程的使用，提高了程序设计的灵活性。存储过程可以使用流程控制语句组织程序结构，方便实现结构较复杂的程序的编写，使设计过程具有很强的灵活性。

② 存储过程把一组功能代码作为单位组件。一旦被创建，存储过程作为一个整体，可以被其他程序多次反复调用。对于数据库程序设计人员，可以根据实际情况，对存储过程进行维护，不会对调用程序产生不必要的影响。

③ 使用存储过程有利于提高程序的执行速度。在数据库操作中，因为存储过程在执行之前已经被预编译，对于包含大量SQL代码或者需要被反复执行的代码段，使用存储过程会大大提高其执行速度。相对于存储过程，批处理的SQL语句段在每次运行之前都要进行编译，导致运行速度较慢。

④ 使用存储过程能减少网络访问的负荷。在访问网络数据库的过程中，如果采用存储过程的方式对SQL语句进行组织，当需要调用存储过程时，仅需在网络中传输调用语句即可，从而大大减少了网络的流量和负载。

⑤ 作为一种安全机制，系统管理员可以充分利用存储过程对相应数据的访问权限进行限制，从而避免非授权用户的非法访问，进一步保证数据访问的安全性。

三、创建和执行存储过程

创建存储过程的语法格式如下：

```
CREATE PROCEDURE sp_name ([proc_parameter[,…]])
  [characteristic…]
  routine_body
```

其中：（1）proc_parameter:[IN|OUT|INOUT]param_name type（2）characteristic:LANGUAGE SQL|[NOT]DETERMINISTIC|{CONTAINS SQL|NO SQL|READS SQL DATA|MODIFIES SQL DATA}|SQLSECURITY{DEFINER|INVOKER}|COMMENT'string'

（1）创建和执行不带输入参数的存储过程

```
CREATE PROCEDURE sp_name()
  BEGIN
    MySQL语句;
  END;
```

【例】在“学生选课”数据库中创建一个名为p_jiaoshi1的存储过程。该存储过程输出teacher表中所有tedu字段为“硕士研究生”的记录。对应的SQL语句如下：

```
DELIMITER //
CREATE PROCEDURE p_jiaoshi1()
 BEGIN
    SELECT * FROM teacher WHERE tedu='硕士研究生';
 END;//
DELIMITER ;
```

注意：

结束符的作用就是告诉MySQL解释器，该段命令已经结束，MySQL可以执行了。默认情况下，结束符是分号（;）。在客户端命令行中，如果有一行命令以分号结束，那么回车后，MySQL将会执行该命令。由于存储过程包含多条语句，并且每条语句以“;”结束，所以必须先改变结束符。改变结束符可以使用DELIMITER语句。例6-1中“DELIMITER//”语句的作用就是把结束符临时改为“//”，存储过程语句输入结束后，再用“DELIMITER;”语句把结束符改回“;”。

存储过程创建成功后，用户就可以执行存储过程了。执行不带参数的存储过程的语法如下：

```
CALL sp_name();
```

【例】执行例6-1中创建的存储过程p_jiaoshi1。对应的SQL语句如下：

```
USE学生选课;
CALL p_jiaoshi1();
```

结果：

```
mysql> CALL p_jiaoshi1();
+------+--------+---------+------------+----------+
| tno  | tname  | tgender | tedu       | tpro     |
+------+--------+---------+------------+----------+
| t002 | 李琦   | 男      | 硕士研究生 | 副教授   |
| t003 | 王艳红 | 女      | 硕士研究生 | 讲师     |
| t005 | 万丽   | 女      | 硕士研究生 | 助理讲师 |
+------+--------+---------+------------+----------+
3 rows in set (0.03 sec)

Query OK, 0 rows affected (0.05 sec)
```

（2）创建和执行带输入参数的存储过程

1. 创建带输入参数的存储过程输入参数是指由调用程序向存储过程传递的参数，在创建存储过程时定义输入参数，在调用存储过程时给出相应的参数值。

【例】在“学生选课”数据库中创建一个名为p_jiaoshi2的存储过程，该存储过程能根据用户给定的学历进行查询并返回teacher表中对应的记录。

对应的SQL命令如下：

```
USE学生选课;
DELIMITER //
CREATE PROCEDURE p_jiaoshi2(IN tedu1 varchar(20))
BEGIN
  SELECT * FROM teacher WHERE tedu=tedu1;
END; //
DELIMITER ;
```

2. 执行带输入参数的存储过程执行带输入参数的存储过程有两种方法：一种是使用变量名传递参数值，另一种是直接传递一个值给参数。

① 使用变量名传递参数值先通过语句SET@parameter_name=value给一个变量设定值，调用存储过程时再用该变量给参数传递值。其语法格式如下：

```
CALL procedure_name ([@parameter_name] [,…n]);
```

【例】用变量名传递参数值的方法执行存储过程p_jiaoshi2，分别查询学历为“本科”和“博士研究生”的记录。对应的SQL语句如下：

```
SET @inspro='本科';
CALL p_jiaoshi2(@inspro);
SET @inspro='博士研究生';
CALL p_jiaoshi2(@inspro);
```

② 按给定表达式值传递参数在执行存储过程的语句中，直接给定参数的值。采用这种方式传递参数值，给定参数值的顺序必须与存储过程中定义的输入变量的顺序一致。其语法格式如下：

```
CALL procedure_name(value1,value2,…)
```

【例】按给定表达式值传递参数的方式执行存储过程p_jiaoshi2，分别查找学历为“本科”和“博士研究生”的记录。对应的SQL语句如下：

```
CALL p_jiaoshi2('本科');
CALL p_jiaoshi2('博士研究生');
```

结果图：

```
mysql> CALL p_jiaoshi2('本科');
+------+-------+---------+------+------+
| tno  | tname | tgender | tedu | tpro |
+------+-------+---------+------+------+
| t001 | 吴亚飞 | 男      | 本科 | 讲师 |
+------+-------+---------+------+------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> CALL p_jiaoshi2('博士研究生');
+------+-------+---------+-----------+------+
| tno  | tname | tgender | tedu      | tpro |
+------+-------+---------+-----------+------+
| t004 | 马志超 | 男      | 博士研究生 | 教授 |
| t007 | 赵楠   | 女      | 博士研究生 | 教授 |
+------+-------+---------+-----------+------+
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

（3）创建和执行带输出参数的存储过程

如果需要从存储过程中返回一个或多个值，可以在创建存储过程的语句中定义输出参数。定义输出参数，需要在CREATE PROCEDURE语句中定义参数时在参数名前面指定OUT关键字。语法格式如下：

```
OUT parameter_name datatype[=default]
```

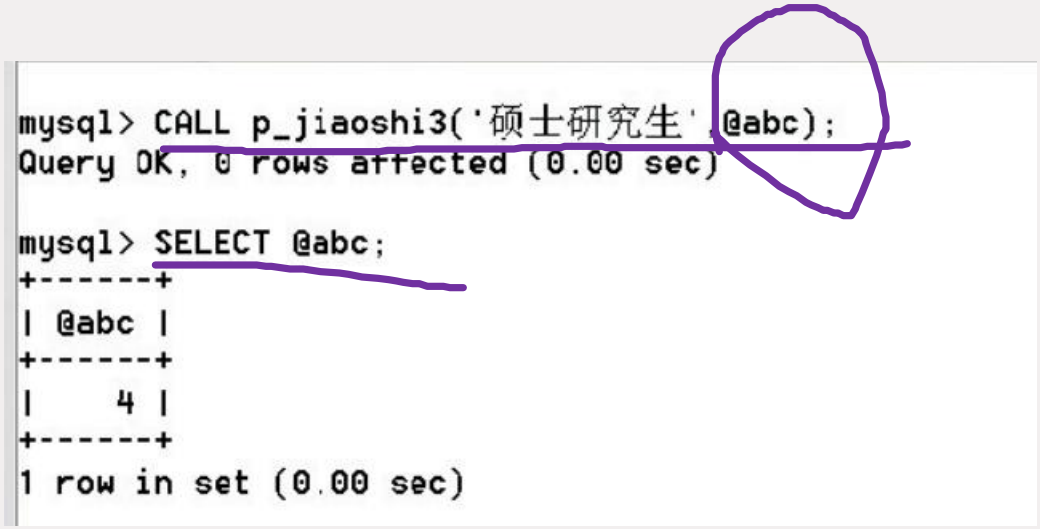
【例】创建存储过程p_jiaoshi3，要求能根据用户给定的学历值，统计出“教师”表的所有教师中学历为该值的教师人数，并将结果以输出变量的形式返回给调用者。对应的SQL语句如下：

```
DELIMITER //
CREATE PROCEDURE p_jiaoshi3(IN tedu1 varchar(20),
                            OUT teachernum smallint)
BEGIN
  SELECT COUNT(*) INTO teachernum FROM teacher WHERE tedu=tedu1;
END//
DELIMITER ;
```

【例】执行存储过程p_jiaoshi3，统计教师表中学历为“硕士研究生”的教师人数。由于在存储过程p_jiaoshi3中使用了输出参数teachernum，所以在调用该存储过程之前，要先设置一个变量来接收存储过程的输出参数。对应的SQL语句如下：

```
SET @abc=0;
CALL p_jiaoshi3('硕士研究生',@abc);
SELECT @abc;
```

结果图：



```
mysql> CALL p_jiaoshi3('硕士研究生',@abc);
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @abc;
+------+
| @abc |
+------+
|    4 |
+------+
1 row in set (0.00 sec)
```

【例】在“学生选课”数据库中创建存储过程p_jiaoshi4，要求能根据用户给定的性别，统计“教师”表中性别为该值的教师人数，并将结果以输出变量的形式返回给用户。对应的SQL语句如下：

```
USE学生选课;
DELIMITER //
CREATE PROCEDURE p_jiaoshi4(in_sex char(2),out out_num int)
BEGIN
  IF in_sex='男' THEN
    SELECT COUNT(tgender) INTO out_num FROM teacher WHERE tgende
r ='男';
  ELSE
    SELECT COUNT(tgender) INTO out_num FROM teacher WHERE tgende
r ='女';
  END IF;
END//
DELIMITER ;
```

【例】执行存储过程p_jiaoshi4，统计教师表中性别为“男”的教师人数。对应的SQL语句如下：

```
SET @abc=0;
CALL p_jiaoshi4('男',@abc);
SELECT @abc;
```

结果：

```
mysql> CALL p_jiaoshi4('男',@abc);
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @abc;
+------+
| @abc |
+------+
|    4 |
+------+
1 row in set (0.00 sec)
```

四、管理存储过程

（1）查看存储过程

存储过程创建后被存储在information_schema数据库的ROUTINES表中，其源代码被存放在系统数据库mysql的proc表中。可以使用以下两种方法来显示数据库内存储过程的列表。

① SELECT name FROM mysql.proc WHERE db='数据库名'。

② SELECT routine_name FROM information_schema. ROUTINES WHERERoutine_schema='数据库名'。使用“SHOW PROCEDURE status WHERE db='数据库名'”可以显示数据库内存储过程的名称和详细信息。

```
mysql> SHOW CREATE PROCEDURE p_jiaoshi3\G
*************************** 1. row ***************************
           Procedure: p_jiaoshi3
            sql_mode: STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION
    Create Procedure: CREATE DEFINER=`root`@`localhost` PROCEDURE `p_jiaoshi3`(IN tedu
OUT teachernum smallint)
BEGIN
SELECT COUNT(*) INTO teachernum FROM teacher WHERE tedu=tedu1;
END
character_set_client: gb2312
collation_connection: gb2312_chinese_ci
  Database Collation: gb2312_chinese_ci
1 row in set (0.00 sec)
```

（2）修改存储过程

修改存储过程是由ALTER PROCEDURE语句来完成的，其语法格式如下：

```
ALTER PROCEDURE sp_name [characteristic…];
```

注意！使用ALTER语句只能修改存储过程的特性。如果要重新定义已有的存储过程，建议先删除该存储过程，然后再进行创建。

以执行。对应的

```
mysql> SELECT SPECIFIC_NAME,SQL_DATA_ACCESS,SECURITY_TYPE
    -> FROM information_schema.Routines
    -> WHERE ROUTINE_NAME='p_jiaoshi1' ;
+---------------+-------------------+---------------+
| SPECIFIC_NAME | SQL_DATA_ACCESS   | SECURITY_TYPE |
+---------------+-------------------+---------------+
| p_jiaoshi1    | MODIFIES SQL DATA | INVOKER       |
+---------------+-------------------+---------------+
1 row in set (0.04 sec)
```

（3）删除存储过程

存储过程的删除是通过DROP PROCEDURE语句来实现的。其语法格式如下：

```
DROP PROCEDURE [IF EXISTS] sp_name;
```

注意：删除时如果存储过程不存在，使用IF EXISTS语句可以防止发生错误。

【例】删除“学生选课”数据库中的存储过程p_jiaoshi2。对应的SQL语句如下：

```
USE学生选课;
DROP PROCEDURE p_jiaoshi2;
```

五、存储函数

（1）存储过程与存储函数的联系与区别

存储函数和存储过程在结构上很相似，都是由SQL语句和过程式语句组成的代码段，都可以被别的应用程序或SQL语句所调用。

但是它们之间是有区别的，主要区别如下：

① 存储函数由于本身就要返回处理的结果，所以不需要输出参数，而存储过程则需要用输出参数返回处理结果。

② 存储函数不需要使用CALL语句进行调用，而存储过程必须使用CALL语句进行调用。

③ 存储函数必须使用RETURN语句返回结果，存储过程不需要RETURN语句返回结果。

（2）创建和执行存储函数

在MySQL中，创建存储函数的基本语法结构如下：

```
CREATE FUNCTION fn_name ([func_parameter[,…]])
RETURNS type
[characteristic…]
routine_body
```

【例】创建一个存储函数，返回两个数中的最大数。

```
DELIMITER //
CREATE FUNCTION maxNumber(x SMALLINT UNSIGNED,
y SMALLINT UNSIGNED)
RETURNS SMALLINT
BEGIN
  DECLARE max SMALLINT UNSIGNED DEFAULT 0;
  IF x>y
    THEN SET max = x;
    ELSE SET max = y;
  END IF;
  RETURN max;
END//
DELIMITER ;
```

调用该存储函数的过程如下：

```
SET @num1=10;
SET @num2=20;
SET @result=maxNumber(@num1,@num2);
SELECT @result;
```

运行结果：

【例】创建

```
DELIMITER //
CREATE FUNCTION addSum()
RETURNS SMALLINT
BEGIN
  DECLARE i INT DEFAULT 0;
  DECLARE sum INT DEFAULT 0;
  Sum_loop:LOOP
                SET i=i+1;
                SET sum=sum+i;
                IF i>=100 THEN LEAVE Sum_loop;
                END IF;
              END LOOP Sum_loop;
RETURN sum;
END//
DELIMITER ;
```

调用该存储函数的过程如下：

```
SET @result=addSum();
SELECT @result;
```

运行结果：

```
mysql> SET @result=addSum();
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @result;
+----------+
| @result |
+----------+
|    5050 |
+----------+
1 row in set (0.00 sec)
```

（3）查看存储函数

同存储过程相同，存储函数被创建之后，用户也可以使用同样的方法来查看用户创建的存储函数的相关信息。可以使用以下两种方法显示数据库内存储函数的列表。

① SELECT name FROM mysql.proc WHERE db='数据库名'。

② SELECT routine_name FROM information_schema. ROUTINES

```
WHERE routine_schema='数据库名'。
```

使用“SHOW PROCEDURE status WHERE db='数据库名'”可以显示数据库内所有存储函数的名称和存储函数的详细信息。使用“SHOW CREATE FUNCTION数据库.存储函数名”可以查看指定存储函数的定义信息。

（4）删除存储函数

存储函数的删除是通过DROP FUNCTION语句来实现的。其语法格式如下：

```
DROP FUNCTION [IF EXISTS] fn_name;
```

例如：删除addSum函数，可以使用如下命令：

```
DROP FUNCTION addSum;
```

（5）MySQL的系数函数

为了能更好地为用户服务，MySQL提供了丰富的系统函数，这些函数无需定义就能直接使用，其中包括数学函数、聚合函数、字符串函数、日期和时间函数等。

1. 数学函数

ABS（x）：返回x的绝对值。

BIN（x）：返回x的二进制值。

CEILING（x）：返回大于等于x的最小整数值。

EXP（x）：返回自然对数e的x次方。

FLOOR（x）：返回小于等于x的最大整数值。

LN（x）：返回x的自然对数。

LOG（x,y）：返回x以y为底的对数。

MOD（x,y）：返回x/y的余数。

PI（）：返回圆周率的值。

RAND（）：返回0到1内的随机数。

ROUND（x,y）：返回x四舍五入到y位小数的值。

SIGN（x）：返回x的符号。其中-1代表负数，1代表正数，0代表0。

SQRT（x）：返回x的平方根。

2. 字符串函数

ASCII（char）：返回字符的ASCII码值。

CONCAT（s1,s2…,sn）：将字符串s1,s2…,sn连接成一个字符串。

LCASE（str）或LOWER（str）：返回将字符串str中所有字符改变为小写字母后的结果。

UCASE（str）或UPPER（str）：返回将字符串str中所有字符改变为大写字母后的结果。

LEFT（str,x）：返回字符串str中最左边的x个字符。

RIGHT（str,x）：返回字符串str中最右边的x个字符。

LENGTH（s）：返回字符串str中的字符数。

LTRIM（str）：从字符串str中去掉开头的空格。

RTRIM（str）：从字符串str中去掉尾部的空格。

TRIM（str）：去除字符串str首部和尾部的所有空格。

POSITION（substr,str）：返回子串substr在字符串str中第一次出现的位置。

REVERSE（str）：返回颠倒字符串str后的结果。

STRCMP（s1,s2）：比较字符串s1和s2的大小，s1大于s2时返回1，s1等于s2时返回0， s1小于s2时返回-1。

3. 日期和时间函数

CURDATE（）或CURRENT_DATE（）：返回系统当前的日期。

CURTIME（）或CURRENT_TIME（）：返回系统当前的时间。

DAYOFWEEK（date）：返回日期date是一个星期的第几天（1～7）。

DAYOFMONTH（date）：返回日期date是一个月的第几天（1～31）。

DAYOFYEAR（date）：返回日期date是一年的第几天（1～366）。

HOUR（time）：返回时间time的小时值（0～23）。

MINUTE（time）：返回时间time的分钟值（0～59）。

MONTH（date）：返回日期date的月份值（1～12）。

MONTHNAME（date）：返回日期date的月份名。

NOW（）：返回系统当前的日期和时间

。

QUARTER（date）：返回日期date在一年中所处的季度（1～4）。

WEEK（date）：返回日期date为一年中第几周（0～53）。

YEAR（date）：返回日期date的年份（1000～9999）。

六、事务

（1）事务概述

所谓的事务是指由用户定义的一系列数据库更新操作，这些操作要么都执行，要么都不执行，是一个不可分割的逻辑工作单元。这里的更新操作主要是指对数据库内容产生修改作用的操作，如INSERT、DELETE、UPDATE等操作。事务是实现数据库中数据一致性的重要技术。例如在银行转账业务的处理过程中，客户A要给客户B转账。当转账进行到一半时，发生断电等异常事故，导致客户A的钱已转出，客户B的钱还没有转入，这样就会导致数据库中数据的不一致，给客户带来损失。在转账业务处理中引入事务机制，就可以在意外发生时撤销整个转账业务，恢复数据库到数据处理之前的状态，从而确保数据的一致性。

1. 事务处理语句

① 启动事务MySQL启动事务的语句格式如下：

```
START TRANSACTION;
```

② 提交事务启动事务之后，就开始执行事务内的SQL语句，当SQL语句执行完毕后，必须提交事务，才能使事务中的所有操作永久生效。提交事务的语句格式如下：

```
COMMIT;
```

③ 回滚事务当事务在执行过程中遇到错误时，事务中的所有操作都要被取消，返回到事务执行前的状态，这就是回滚事务。回滚事务的语句格式如下：

```
ROLLBACK;
```

2. 事务的特性事务必须具有ACID特性，即原子性（Atomicity ）、一致性（Consistency ）、隔离性（Isolation）和持久性（Durability）。

① 原子性原子性是指事务是一个不可分割的逻辑工作单元，事务处理的操作要么全部执行，要么全部不执行。

② 一致性一致性是指事务在执行前后必须处于一致性状态。如果事务全部正确执行，数据库的变化将生效，从而处于有效状态；如果事务执行失败，系统将会回滚，从而将数据库恢复到事务执行前的有效状态。

③ 隔离性隔离性是指当多个事务并发执行时，各个事务之间不能相互干扰。

④ 持久性持久性是指事务完成后，事务对数据库中数据的修改将永久保存。

（2）事务的提交

为了使事务中SQL语句执行的修改操作永久保存在数据库中，事务处理结束时必须由用户提交事务。例如，在手动提交的方式下，启动一个事务，在teacher表中插入两条记录，具体语句如下：

```
START TRANSACTION;
INSERT INTO teacher
VALUES('t006','张君瑞','男','硕士研究生','副教授');
INSERT INTO teacher
VALUES('t007','赵楠','女','博士研究生','教授');
```

执行之后用SELECT语句查询teacher表。执行结果如上图：

```
mysql> select * from teacher;
+------+--------+---------+------------+----------+
| tno  | tname  | tgender | tedu       | tpro     |
+------+--------+---------+------------+----------+
| t001 | 吴亚飞 | 男      | 本科       | 讲师     |
| t002 | 李琦   | 男      | 硕士研究生 | 副教授   |
| t003 | 王艳红 | 女      | 硕士研究生 | 讲师     |
| t004 | 马志超 | 男      | 博士研究生 | 教授     |
| t005 | 万丽   | 女      | 硕士研究生 | 助理讲师 |
| t006 | 张君瑞 | 男      | 硕士研究生 | 副教授   |
| t007 | 赵楠   | 女      | 博士研究生 | 教授     |
+------+--------+---------+------------+----------+
7 rows in set (0.00 sec)
```

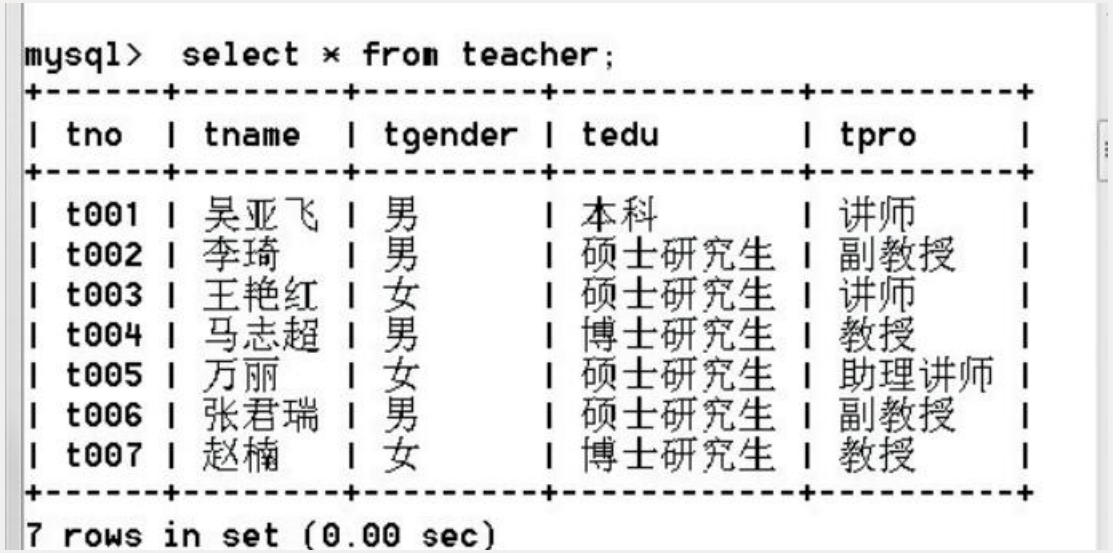
从以上结果来看，似乎已经完成了事务的处理，但是退出数据库重新登录后，再次对teacher表进行查询。结果如图：

```
mysql> use 学生选课
Database changed
mysql>  select * from teacher;
+------+--------+--------+------------+----------+
| tno  | tname  | tgender | tedu       | tpro     |
+------+--------+--------+------------+----------+
| t001 | 吴亚飞 | 男     | 本科       | 讲师     |
| t002 | 李琦   | 男     | 硕士研究生 | 副教授   |
| t003 | 王艳红 | 女     | 硕士研究生 | 讲师     |
| t004 | 马志超 | 男     | 博士研究生 | 教授     |
| t005 | 万丽   | 女     | 硕士研究生 | 助理讲师 |
+------+--------+--------+------------+----------+
5 rows in set (0.00 sec)
```

```
START TRANSACTION;
INSERT INTO teacher
    VALUES('t006','张君瑞','男','硕士研究生','副教授');
INSERT INTO teacher
    VALUES('t007','赵楠','女','博士研究生','教授');
COMMIT;
```

从以上结果可以看出，事务中的记录插入操作最终并未完成，这是因为事务未经提交（COMMIT）就已经退出数据库了，由于采用的是手动提交模式，事务中的操作被自动取消了。为了能够把两条记录永久写入数据库中，需要在事务处理结束后加入COMMIT语句来完成整个事务的提交。具体代码如上：

执行完毕后，退出数据库重新登录，使用SELECT语句查询teacher表中的记录，查询结果如图·



```
mysql> select * from teacher;
+------+--------+---------+------------+----------+
| tno  | tname  | tgender | tedu       | tpro     |
+------+--------+---------+------------+----------+
| t001 | 吴亚飞 | 男      | 本科       | 讲师     |
| t002 | 李琦   | 男      | 硕士研究生 | 副教授   |
| t003 | 王艳红 | 女      | 硕士研究生 | 讲师     |
| t004 | 马志超 | 男      | 博士研究生 | 教授     |
| t005 | 万丽   | 女      | 硕士研究生 | 助理讲师 |
| t006 | 张君瑞 | 男      | 硕士研究生 | 副教授   |
| t007 | 赵楠   | 女      | 博士研究生 | 教授     |
+------+--------+---------+------------+----------+
7 rows in set (0.00 sec)
```

（3）事务的回滚

如果事务尚未提交出现了操作错误，可以通过事务的回滚来取消当前事务，把数据库恢复到事务处理之前的状态。

（4）事务的隔离级别

MySQL在数据库访问过程中采用的是并发访问方式。在多个线程同时开启事务访问数据库时，可能会出现脏读、不可重复读以及幻读等情况。

① 脏读

脏读就是一个事务读取了另一个事务没有提交的数据。即第一个事务正在访问数据，并且对数据进行了修改，当这些修改还没有提交时，第二个事务访问和使用了这些数据。如果第一个事务回滚，那么第二个事务访问和使用的数据就是错误的脏数据。

② 不可重复读

不可重复读是指在一个事务内，对同一数据进行了两次相同查询，但返回结果不同。这是由于在一个事务两次读取数据之间，有第二个事务对数据进行了修改，造成两次读取数据的结果不同。

③ 幻读

幻读是指在同一事务中，两次按相同条件查询到的记录不一样。造成幻读的原因在于事务处理没有结束时，其他事务对同一数据集合增加或者删除了记录。为了避免以上情况的发生，MySQL设置了事务的四种隔离级别，由低到高分别为READ UNCOMMITTED、READ COMMITTED、REPEATABLE READ、SERIALIZABLE，能够有效地防止脏读、不可重复读以及幻读等情况。

READ UNCOMMITTED是指“读未提交”，该级别下的事务可以读取另一个未提交事务的数据，它是最低事务隔离级别。这种隔离级别在实际应用中容易出现脏读等情况，因此很少被应用。

READ COMMITTED是指“读提交”，该级别下的事务只能读取其他事务已经提交的数据。这种隔离级别容易出现不可重复读的问题。

REPEATABLE READ是指“可重复读”，是MySQL的默认事务隔离级别。它确保同一事务的多个实例并发读取数据时，读到的数据是相同的。这种隔离级别容易出现幻读的问题。

SERIALIZABLE是指“可串行化”，是MySQL最高的事务隔离级别。它通过对事务进行强制性的排序，使事务之间不会相互冲突，从而解决幻读问题。但是这种隔离级别容易出现超时现象和锁竞争。

各个隔离级别可能产生的问题：

隔离级别	脏读	不可重复读	幻读
READ UNCOMMITTED	√	√	√
READ COMMITTED	×	√	√
REPEATABLE READ	×	×	√
SERIALIZABLE	×	×	×

用户可以用SET TRANSACTION语句改变当前会话或所有新建连接的隔离级别。它的语法格式如下：

```
    SET [SESSION | GLOBAL] TRANSACTION ISOLATION LEVEL {READ UNCOMMI
TTED | READ
COMMITTED | REPEATABLE READ | SERIALIZABLE}
```

例如设置当前会话的隔离级别为READ COMMITTED，具体语句如下：

```
    SET  SESSION  TRANSACTION  ISOLATION  LEVEL  READ COMMITTED;
```

本章小结

- 变量的定义与赋值。
- 流程控制语句。
- 光标的定义与使用。
- 存储过程的创建、调用、修改和删除。
- 存储函数的创建、使用、查看和删除。
- 事务的特性，启动事务、提交事务、回滚事务的语句。
- 事务的隔离级别。

实例练习1：在“网上书店”数据库中创建存储过程

目的：掌握存储过程的创建和执行.

内容：在“网上书店”数据库中创建一个名为proc_1的存储过程，实现查询所有会员信息的功能.

实例练习2：在“网上书店”数据库中创建带输入输出参数的存储过程

目的：掌握存储过程中输入、输出参数的使用。

内容：

（1）在“网上书店”数据库中创建一个名为proc_2的存储过程，要求实现如下功能：根据会员昵称查询会员的积分情况。并调用存储过程，查询“平平人生”和“感动心灵”的积分。

（2）在“网上书店”数据库中创建一个名为proc_3的存储过程，要求实现如下功能：根据会员昵称查询会员的订购信息，如果该会员没有订购任何图书，则输出“某某会员没有订购图书”的信息；否则输出订购图书的相关信息。调用存储过程，显示会员“四十不惑”订购图书的情况。

实例练习3：在“网上书店”数据库中实现事务处理

目的：掌握事务的启动、提交和回滚

内容：启动一个事务，在事务中使用SQL语句删除“网上书店”数据库中会员表的所有记录，第一次不提交事务，第二次提交事务，第三次回滚事务。重启MySQL服务器分别查看记录是否被永久删除。

课后小习题：

一. 选择题

1. CREATE PROCEDURE是用来创建（ ）的语句。

A. 程序　　B. 存储过程　C. 触发器　D. 存储函数

2. 要删除一个名为AA的存储过程，应该使用命令（ ）PROCEDURE AA。

A. DELETE　B. ALTER　　C. DROP　　D. EXECUTE

3. 执行带参数的存储过程，正确的方法为：（ ）

A. CALL存储过程名（参数）

B. CALL存储过程名参数

C. 存储过程名=参数

D. 以上答案都正确

4. 用户定义的一系列数据库更新操作，这些操作要么都执行，要么都不执行，是一个不可分割的逻辑工作单元，这体现了事务的（ ）。

A. 原子性　B. 一致性　C. 隔离性　D. 持久性

5. 事务的隔离级别中，（ ）可以解决幻读问题。

A. READ UNCOMMITTED

B. READ COMMITTED

C. REPEATABLE READ

D. SERIALIZABLE

二.问答题

1. 什么是存储过程？写出存储过程的创建、修改和删除语句。

2. 什么是存储函数？写出存储函数的创建、查看和删除语句。

3. 事务具有哪些特性?

4. 事务的隔离级别有哪些，各自有什么特点?

THANKS

感谢观看

微信公众号：灰灰考研