# 11.MySQL高级开发上（库层次下的操作对象–存储过程应用）√

## 1.存储过程基础应用

### 1.0 介绍

**是mysql数据库库层面下的操作对象，和表，视图，函数，触发器，事件同等级**

> 1.存储过程（Stored Procedure）是一种在数据库中存储复杂程序，以便外部程序调用的一种数据库对象。
> 2.存储过程是为了完成特定功能的SQL语句集，经编译创建并保存在数据库中，用户可通过指定存储过程的名字并给定参数(需要时)来调用执行。
> 3.存储过程思想上很简单，就是数据库 SQL 语言层面的代码封装与重用。

**优点：**

- 存储过程可封装，并隐藏复杂的商业逻辑。
- 存储过程可以回传值，并可以接受参数。
- 存储过程无法使用 SELECT 指令来运行，因为它是子程序，与查看表，数据表或用户定义函数不同。
- 存储过程可以用在数据检验，强制实行商业逻辑等。

**缺点：**

- 存储过程，往往定制化于特定的数据库上，因为支持的编程语言不同。当切换到其他厂商的数据库系统时，需要重写原有的存储过程。
- 存储过程的性能调校与撰写，受限于各种数据库系统。


### 1.1 创建存储过程的语法

**在mysql客户端使用这种语法逻辑，如果使用第三方软件连接，本身就定义好，不需要严格遵守语句。**

- 存储过程就是具有名字的一段代码，用来完成一个特定的功能。
- 创建的存储过程保存在数据库的数据字典中（元数据 information_schema.ROUTINES）。

```
DELIMITER $$            --->                先将语句结束符号改写为$$
CREATE
    [DEFINER = { user | CURRENT_USER }] --->  定义者。存储过程的定义用户，默认是创建存储过程的用户  （备份恢复要保证定义用户
    PROCEDURE `world`.`P_test`          --->  `那个库`.`创建存储过程的名`
    proc_parameter:                     --->  参数列表
   [ 空 |IN | OUT | INOUT ] param_name type  [参数模式]参数名 参数类型

    BEGIN               --->                标记存储过程开始
       过程体           --->                1.sql语句的叠加 2.流程控制结构 3.循环结构
     END$$              --->                标记存储过程结束

DELIMITER ;                                 最后语句完成之后，再将语句结束符改为默认的;

语法解释:
DELTMITER：是用来定义语句结束标记的
PROCEDURE：不进行参数类型的定义（空）
参数模式 ：
IN ：  输入参数，单独传参。
OUT ：  输出参数，作为返回值的参数。
INOUT：  既输入有输出，可做输入也可做输出。
```

## 1.2 调用存储过程

```
mysql> CALL 存储过程名(实参列表);
```

## 1.3 存储过程创建实列

### 1.3.1 空参数列表应用

Bash | Copy

```
0.创建空参数列表的存储过程
DELIMITER $$

CREATE
PROCEDURE `world`.`p_null`()     ()参数列表为空

BEGIN
select * from world.city where countrycode='CHN';
END$$

DELIMITER ;
2.查看创建的存储过程  (information_schema)
                                    库名.表名              列名
mysql>  select * from  information_schema.routines where routine_schema='world'\G;
*************************** 1. row ***************************
           SPECIFIC_NAME: p_null
         ROUTINE_CATALOG: def
          ROUTINE_SCHEMA: world
            ROUTINE_NAME: p_null
            ROUTINE_TYPE: PROCEDURE
               DATA_TYPE:
CHARACTER_MAXIMUM_LENGTH: NULL
  CHARACTER_OCTET_LENGTH: NULL
       NUMERIC_PRECISION: NULL
           NUMERIC_SCALE: NULL
      DATETIME_PRECISION: NULL
      CHARACTER_SET_NAME: NULL
          COLLATION_NAME: NULL
          DTD_IDENTIFIER: NULL
            ROUTINE_BODY: SQL
      ROUTINE_DEFINITION: BEGIN
select * from world.city where countrycode='CHN';
END
           EXTERNAL_NAME: NULL
       EXTERNAL_LANGUAGE: SQL
         PARAMETER_STYLE: SQL
        IS_DETERMINISTIC: NO
         SQL_DATA_ACCESS: CONTAINS SQL
                SQL_PATH: NULL
           SECURITY_TYPE: DEFINER
                 CREATED: 2021-04-10 20:27:11
            LAST_ALTERED: 2021-04-10 20:27:11
                SQL_MODE:
         ROUTINE_COMMENT:
                 DEFINER: oldguo@10.0.0.%
    CHARACTER_SET_CLIENT: utf8
    COLLATION_CONNECTION: utf8_general_ci
      DATABASE_COLLATION: utf8mb4_0900_ai_ci
3.调用我们创建的存储过程
mysql> call world.p_null();
```

```
+------+--------------------+-------------+----------------+------------+
| ID   | Name               | CountryCode | District       | Population |
+------+--------------------+-------------+----------------+------------+
| 1890 | Shanghai           | CHN         | Shanghai       |    9696300 |
| 1891 | Peking             | CHN         | Peking         |    7472000 |
| 1892 | Chongqing          | CHN         | Chongqing      |    6351600 |
| 1893 | Tianjin            | CHN         | Tianjin        |    5286800 |
| 1894 | Wuhan              | CHN         | Hubei          |    4344600 |
| 1895 | Harbin             | CHN         | Heilongjiang   |    4289800 |
| 1896 | Shenyang           | CHN         | Liaoning       |    4265200 |
| 1897 | Kanton [Guangzhou] | CHN         | Guangdong      |    4256300 |

```

### 1.3.2 IN 参数列表应用

**输入参数到存储过程，调用存储过程的sql语句，一个存储过程只能传单个参数。**

Bash | Copy

```
0.创建in参数列表的存储过程
DELIMITER $$

CREATE
PROCEDURE `world`.`p_in`(in cc char(64))   ()参数列表为in，要设置数据类型，比原表的更长

BEGIN
select * from world.city where countrycode=cc;  调用存储过程cc
END$$

DELIMITER ;

1. 调用存储过程，进程传参
mysql> CALL p_in('CHN');
+------+--------------------+-------------+----------------+------------+
| ID   | Name               | CountryCode | District       | Population |
+------+--------------------+-------------+----------------+------------+
| 1890 | Shanghai           | CHN         | Shanghai       |    9696300 |
| 1891 | Peking             | CHN         | Peking         |    7472000 |
| 1892 | Chongqing          | CHN         | Chongqing      |    6351600 |
| 1893 | Tianjin            | CHN         | Tianjin        |    5286800 |
| 1894 | Wuhan              | CHN         | Hubei          |    4344600 |
| 1895 | Harbin             | CHN         | Heilongjiang   |    4289800 |
| 1896 | Shenyang           | CHN         | Liaoning       |    4265200 |
| 1897 | Kanton [Guangzhou] | CHN         | Guangdong      |    4256300 |
```

### 1.3.3 OUT 参数列表应用

输出参数，作为返回值的参数。

Bash | Copy

```
0.创建out参数列表的存储过程
DELIMITER $$

CREATE
PROCEDURE `world`.`p_out`(out c int)   ()参数列表为out，要设置数据类型，比原表的更长

BEGIN
select count(*) from world.city into c    将sql语句查询结果赋值给c
select c;       因为存储过程默认不会输出，所有需要调用c（局部变量不用@符号就可以调用）
END$$

DELIMITER ;
1.调用存储过程中输出的局部变量，需要先在外部定义用户变量。将局部变量的值赋值给用户变量，再调用用户变量显示值。
1.1 定义用户变量
mysql> SET @a:=0;
1.2 将局部变量的值赋值给用户变量，调用用户变量显示值。
mysql> call world.p_out(@a);
+------+
| c    |
+------+
| 4079 |
+------+

```

### 1.3.4 in和out 混合使用

Bash | Copy

```
题目：分别统计每个国家的人口数量 传参那个国家（in），返回国家的国家id和总人口数(out)
0.创建in out 混合参数列表的存储过程
DELIMITER $$

CREATE
PROCEDURE `world`.`p_in_out`(in cc char(64),out c char(64),out s int)  cc输入参数列表国家 c,s输出参数列表（国家id）(

BEGIN
select countrycode,sum(population) from world.city
where countrycode=cc
group by cc
into c,s;
select c,s;      因为存储过程默认不会输出，所有需要调用c,s（局部变量不用@符号就可以调用）
END$$
DELIMITER ;

1.调用存储过程中输出的局部变量，需要先在外部定义用户变量。将局部变量的值赋值给用户变量，再调用用户变量显示值。
1.1 定义用户变量
mysql> SET @a:=0;
mysql> SET @b:=0;
1.2 将局部变量的值赋值给用户变量，调用用户变量显示值。再输入参数
mysql> CALL world.p_in_out('CHN',@a,@b);
+------+-----------+
| c    | s         |
+------+-----------+
| CHN  | 175953614 |
+------+-----------+
```

### 1.3.5 INOUT 参数列表应用

**INOUT： 既输入有输出，可做输入也可做输出。**

```
题目：分别统计每个国家的人口数量 传参那个国家并且返回国家的国家id（inout） 和总人口数(out)
0.创建inout参数列表的存储过程
DELIMITER $$

CREATE
PROCEDURE `world`.`p_inout`(inout cc char(64),out s int)  cc inout参数列表国家 s输出参数列表(国家总人口)

BEGIN
select countrycode,sum(population) from world.city
where countrycode=cc
group by cc
into cc,s;
select cc,s;      因为存储过程默认不会输出，所有需要调用cc,s（局部变量不用@符号就可以调用）
END$$
DELIMITER ;
1.调用存储过程中输出的局部变量，需要先在外部定义用户变量。将局部变量的值赋值给用户变量，再调用用户变量显示值。
1.1 定义用户变量
mysql> SET @a:=0;
mysql> SET @b:=0;
1.2 将局部变量的值赋值给用户变量，调用用户变量显示值。再输入参数
mysql> CALL world.p_in_out('CHN',@a,@b);
+------+-----------+
| c    | s         |
+------+-----------+
| CHN  | 175953614 |
+------+-----------+
```

# 2.存储过程中变量(局部变量)的使用

## 2.1 局部变量使用的套路

> 1.声明：（两种语法方式）
> 方式一：declare 变量名 类型
> 方式二：declare 变量名 类型 default 默认值
> 2.赋值（两种语法方式）
> 方式一：set var=值
> 方式二：select into
> 3.调用
> select 变量名

## 2.2 局部变量使用练习

Bash | Copy

```
练习一
题目要求：创建个存储过程，向指定的表中插入一行随机的值
uname:6字符随机长度。
pass：12位随机密码，第一位是大写，剩下的是随机数字字母组合。
0.首先模拟环境，创建被插入的表
mysql> use world
mysql> create table t1( id int not null primary key auto_increment,
uname varchar(64) not null ,
pass varchar(20) not null )engine=innodb charset=utf8mb4;
1.创建存储过程
DELIMITER $$

CREATE
PROCEDURE `world`.`p_1`()

BEGIN
DECLARE v_u VARCHAR(16);              声明局部变量
DECLARE v_p VARCHAR(64);              声明局部变量
SELECT SUBSTR('abcdefghijklmnopqrstuvwxyz',1+FLOOR(RAND()*21),6) INTO v_u;  赋值v_u
SELECT CONCAT(SUBSTR('ABCDEFGHIJKLMNOPQRSTUVWXYZ',1+FLOOR(RAND()*26),1),
SUBSTR(REPLACE(UUID(),'-',''),1+FLOOR(RAND()*22),11)) INTO V_P;              赋值v_p
INSERT t1(uname,pass) VALUES(v_u,v_p);
END$$
DELIMITER ;
2.调用存储过程，对t1表进行插入一行数据
mysql> call world.p_1();
Query OK, 1 row affected (0.00 sec)

mysql> select * from t1;
+----+--------+--------------+
| id | uname  | pass         |
+----+--------+--------------+
|  1 | cdefgh | P1eba190000c |
+----+--------+--------------+
```

Bash | Copy

```
练习二
题目要求：
uname:随机6位字符串
age:18到25岁
gender :m或f
telnum 以数字1开头 第二位3-9 后面9位000000000-999999999
0.首先模拟环境，创建被插入的表
mysql> use world
mysql> create table t1( id int not null primary key auto_increment,
uname varchar(64) not null ,
age tinyint not null,
gender char(1) not null,
telnum char(11) not null,
udate datetime not null default now() )engine=innodb charset=utf8mb4;
1.创建存储过程
DELIMITER $$
CREATE
    PROCEDURE `world`.`p_2`()
    BEGIN
    DECLARE v_u VARCHAR(64);
    DECLARE v_a TINYINT;
    DECLARE v_g CHAR(1);
    DECLARE v_t CHAR(11);
    SELECT SUBSTR('abcdefghijklmnopqrstuvwxyz',1+FLOOR(RAND()*21),6) INTO v_u;
    SELECT 18+FLOOR(RAND()*8) INTO v_a;
    SELECT SUBSTR('MF',1+FLOOR(RAND()*2),1) INTO v_g;
    SELECT CONCAT('1',3+FLOOR(RAND()*7),LPAD(FLOOR(RAND()*1000000000),9,'0')) INTO v_t;
    INSERT INTO world.t1(uname,age,gender,telnum)
    VALUES (v_u,v_a,v_g,v_t);
    END$$
DELIMITER ;
2.调用存储过程
mysql> call world.p_2();
Query OK, 1 row affected (0.00 sec)

mysql> select * from t1;
+----+--------+-----+--------+-------------+---------------------+
| id | uname  | age | gender | telnum      | udate               |
+----+--------+-----+--------+-------------+---------------------+
|  1 | efghij |  19 | M      | 16398396511 | 2021-04-12 18:50:18 |
+----+--------+-----+--------+-------------+---------------------+
```

# 3.存储过程高级应用–流程控制结构

## 3.1 流程控制结构的分类

> 1.顺序结构：从上到下顺序执行
> 2.分支结构：多条路径选择其中一条路径执行

> 3.循环结构：满足条件，重复执行一段代码

## 3.2 分支结构的分类

### 3.2.1 IF

```
if 条件1 then 语句1;
elseif 条件2 then 语句2;
...;
else 语句n;
end if;
```

### 3.2.2 CASE

```
方法1：等值判断
case 变量|表达式|字段
when 判断的值 then 结果或语句
when 判断的值 then 结果或语句
...
else 结果或语句
end case;

方法2：  多条件判断
case when 条件1 then 语句;
when 条件2 then 语句;
else 语句
end case;
```

### 3.2.3 分支结构的应用

Bash | Copy

```
练习一
题目要求：输入整数体重，判断身材形态。
0.创建存储过程p_if
DELIMITER $$
CREATE
PROCEDURE `world`.`p_if`(IN tz INT)
BEGIN
       DECLARE result VARCHAR(10);
       IF tz <100 THEN SET result='瘦';
       ELSEIF tz BETWEEN 100 AND 130 THEN SET result='壮';
       ELSE
       SET result='胖';
       END IF;
       SELECT result;
    END$$

DELIMITER ;
1.调用存储过程，传入参数
mysql> call world.p_if(120);
+--------+
| result |
+--------+
| 壮     |
+--------+

改写为case语句
0.创建存储过程
DELIMITER $$
CREATE
PROCEDURE `world`.`p_case`(IN tz INT)
BEGIN
       DECLARE result VARCHAR(10);
       CASE
       WHEN tz<100 THEN
       SET result='瘦';
       WHEN tz BETWEEN 100 AND 130 THEN
```

Bash | Copy

```
练习二
题目要求：判断年龄范围
0.创建存储过程
DELIMITER $$
CREATE
PROCEDURE `world`.`p_case1`(IN age INT)
BEGIN
       DECLARE result VARCHAR(10);
       CASE
       WHEN age BETWEEN 0 AND 10 THEN
       SET result='黄口小儿';
       WHEN age BETWEEN 11 AND 20 THEN
       SET result='青少年';
       WHEN age BETWEEN 21 AND 30 THEN
       SET result='青年';
       WHEN age BETWEEN 31 AND 50 THEN
       SET result='中年';
       ELSE
       SET result='退休了';
       END CASE;
       SELECT result;
    END$$
DELIMITER ;
1.调用存储过程
mysql> CALL p_case1(23);
+--------+
| result |
+--------+
| 青年   |
+--------+
```

Bash | Copy

```
练习三
题目要求：判断输入的用户及用户密码信息是否正确，当有一项输入错误，就报错。
0.模拟环境，创建用户密码表
mysql>  create table t1(id int,username varchar(20), pass varchar(20));
mysql> insert into t1 values(1,'a','a'),(2,'b','b');
mysql> select * from t1;
+------+----------+------+
| id   | username | pass |
+------+----------+------+
|    1 | a        | a    |
|    2 | b        | b    |
+------+----------+------+
1.创建存储过程
DELIMITER $$
CREATE
    PROCEDURE `world`.`p_case2`(IN u VARCHAR(20),IN p VARCHAR(20))
    BEGIN
    DECLARE result VARCHAR(20);
    DECLARE COUNT INT DEFAULT 0;
    SELECT COUNT(*)  FROM world.t1 WHERE t1.username=u AND t1.pass=p INTO COUNT;
    CASE WHEN COUNT>0 THEN SET result='success!';
    ELSE
    SET result='error!';
    END CASE;
    SELECT result;
    END$$

DELIMITER ;
2.调用存储过程
mysql> CALL p_case2('a','a');
+----------+
| result   |
+----------+
| success! |
+----------+
mysql> CALL p_case2('a','b');
```

## 3.3 循环结构的分类

> while
> loop
> repeat

### 3.3.1 while

介绍

> 满足条件，则执行循环

语法

```bash
[标签：] while 条件 do
循环体；
end while [标签]；
```

while实列

```bash
题目：向表中插入100行数据，循环100次
0.模拟环境创建表
SELECT * FROM  t1;
CREATE TABLE t1( id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
uname VARCHAR(64) NOT NULL ,
age TINYINT NOT NULL,
gender CHAR(1) NOT NULL,
telnum CHAR(11) NOT NULL,
udate DATETIME NOT NULL DEFAULT NOW() )ENGINE=INNODB CHARSET=utf8mb4;
1.创建存储过程
DELIMITER $$
CREATE
    PROCEDURE `world`.`p_while`()
    BEGIN
     DECLARE v_u VARCHAR(64);
     DECLARE v_a TINYINT;
     DECLARE v_g CHAR(1);
     DECLARE v_t CHAR(11);
     DECLARE i INT DEFAULT 0;
      WHILE i<100
      DO
       SELECT SUBSTR('abcdefghijklmnopqrstuvwxyz',1+FLOOR(RAND()*21),6) INTO v_u;
       SELECT 18+FLOOR(RAND()*8) INTO v_a;
       SELECT SUBSTR('MF',1+FLOOR(RAND()*2),1) INTO v_g;
       SELECT CONCAT('1',3+FLOOR(RAND()*7),LPAD(FLOOR(RAND()*1000000000),9,'0')) INTO v_t;
       INSERT INTO world.t1(uname,age,gender,telnum)
       VALUES (v_u,v_a,v_g,v_t);
      SET i=i+1;
      END WHILE;
    END$$
DELIMITER ;
2.调用存储过程，再查看。
call world.p_while();
```

### 3.3.2 loop

### 介绍

> 死循环

### 语法

```bash
[标签: ] loop
循环体;
end loop [标签];
```

### loop实列+配合leave使用

Bash | Copy

```
题目：向表中插入100行数据，循环100次
0.模拟环境创建表
SELECT * FROM  t1;
CREATE TABLE t1( id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
uname VARCHAR(64) NOT NULL ,
age TINYINT NOT NULL,
gender CHAR(1) NOT NULL,
telnum CHAR(11) NOT NULL,
udate DATETIME NOT NULL DEFAULT NOW() )ENGINE=INNODB CHARSET=utf8mb4;
1.创建存储过程
DELIMITER $$
CREATE
    PROCEDURE `world`.`p_loop`()
    BEGIN
     DECLARE v_u VARCHAR(64);
     DECLARE v_a TINYINT;
     DECLARE v_g CHAR(1);
     DECLARE v_t CHAR(11);
     DECLARE i INT DEFAULT 1;
     lab1:LOOP
       SELECT SUBSTR('abcdefghijklmnopqrstuvwxyz',1+FLOOR(RAND()*21),6) INTO v_u;
       SELECT 18+FLOOR(RAND()*8) INTO v_a;
       SELECT SUBSTR('MF',1+FLOOR(RAND()*2),1) INTO v_g;
       SELECT CONCAT('1',3+FLOOR(RAND()*7),LPAD(FLOOR(RAND()*1000000000),9,'0')) INTO v_t;
      IF i>100
      THEN LEAVE lab1;
      ELSE
       INSERT INTO world.t1(uname,age,gender,telnum)
       VALUES (v_u,v_a,v_g,v_t);
      SET i=i+1;
      END IF;
     END LOOP lab1;

 END$$
DELIMITER ;
2.调用存储过程
CALL p_loop();
3.改写插入多少行存储过程
DELIMITER $$
CREATE
    PROCEDURE `world`.`p_loop1`(IN num INT)
    BEGIN
     DECLARE v_u VARCHAR(64);
     DECLARE v_a TINYINT;
     DECLARE v_g CHAR(1);
     DECLARE v_t CHAR(11);
     DECLARE i INT DEFAULT 1;
     lab1:LOOP
       SELECT SUBSTR('abcdefghijklmnopqrstuvwxyz',1+FLOOR(RAND()*21),6) INTO v_u;
       SELECT 18+FLOOR(RAND()*8) INTO v_a;
```

```
        SELECT SUBSTR('MF',1+FLOOR(RAND()*2),1) INTO v_g;
        SELECT CONCAT('1',3+FLOOR(RAND()*7),LPAD(FLOOR(RAND()*1000000000),9,'0')) INTO v_t;
       IF i>num
       THEN LEAVE lab1;
       ELSE
        INSERT INTO world.t1(uname,age,gender,telnum)
        VALUES (v_u,v_a,v_g,v_t);
       SET i=i+1;
       END IF;
      END LOOP lab1;

 END$$
DELIMITER ;
4.传参调用
CALL p_loop1(50);
```

### 3.3.3 repeat

#### 介绍

> 先执行循环，不满足条件，退出执行

#### 语法

Bash | Copy

```
[标签：] repeat
循环体；
until 条件
end repeat [标签];
```

#### repeat实列

Bash | Copy

```
题目：向表中插入100行数据，循环100次
0.模拟环境创建表
SELECT * FROM  t1;
CREATE TABLE t1( id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
uname VARCHAR(64) NOT NULL ,
age TINYINT NOT NULL,
gender CHAR(1) NOT NULL,
telnum CHAR(11) NOT NULL,
udate DATETIME NOT NULL DEFAULT NOW() )ENGINE=INNODB CHARSET=utf8mb4;
1.创建存储过程
DELIMITER $$
CREATE
    PROCEDURE `world`.`p_repeat`()
    BEGIN
     DECLARE v_u VARCHAR(64);
     DECLARE v_a TINYINT;
     DECLARE v_g CHAR(1);
     DECLARE v_t CHAR(11);
     DECLARE i INT DEFAULT 0;
      REPEAT
       SELECT SUBSTR('abcdefghijklmnopqrstuvwxyz',1+FLOOR(RAND()*21),6) INTO v_u;
       SELECT 18+FLOOR(RAND()*8) INTO v_a;
       SELECT SUBSTR('MF',1+FLOOR(RAND()*2),1) INTO v_g;
       SELECT CONCAT('1',3+FLOOR(RAND()*7),LPAD(FLOOR(RAND()*1000000000),9,'0')) INTO v_t;
       INSERT INTO world.t1(uname,age,gender,telnum)
       VALUES (v_u,v_a,v_g,v_t);
      SET i=i+1;
      UNTIL i>99
      END REPEAT;
    END$$
DELIMITER ;
2.调用存储过程，再查看。
call world.p_repeat();
```

### 3.3.4 循环控制

> iterate ----> continue(继续执行)
> leave -----> break（结束执行）

# 4.存储过程高级应用–游标

## 4.1 游标介绍

Bash | Copy

```
# 什么是游标?
保存select语句的数据集，主要用于对数据集逐行进行处理。
# 使用方法 需要的操作:
定义游标 :
DECLARE cur_1 CURSOR FOR SELECT id,name FROM city;
定义游标异常处理:
declare done int default 1; （定义变量必须放在定义游标之前）
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 0;
打开游标:
open cur_1;
提取数据:
fetch cur_1 into c_id,c_name;
关闭游标:
close cur_1;
```

## 4.2 游标案例

Bash | Copy

```
0.  模拟环境，创建表
SELECT * FROM  t1;
CREATE TABLE t1( id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
uname VARCHAR(64) NOT NULL ,
age TINYINT NOT NULL,
gender CHAR(1) NOT NULL,
telnum CHAR(11) NOT NULL,
udate DATETIME NOT NULL DEFAULT NOW() )ENGINE=INNODB CHARSET=utf8mb4;

插入三行数据，可以用我们之前创建的存储过程
mysql> call p_loop1(3);
mysql> SELECT * FROM t1;
+----+--------+-----+--------+-------------+---------------------+
| id | uname  | age | gender | telnum      | udate               |
+----+--------+-----+--------+-------------+---------------------+
|  1 | hijklm |  18 | M      | 13667132062 | 2021-04-12 22:56:31 |
|  2 | ghijkl |  21 | M      | 17712172916 | 2021-04-12 22:56:31 |
|  3 | rstuvw |  18 | F      | 18341998951 | 2021-04-12 22:56:31 |
+----+--------+-----+--------+-------------+---------------------+
1.创建存储过程
DELIMITER $$
CREATE
    PROCEDURE `world`.`p_c`()
    BEGIN
DECLARE v_i INT;
DECLARE v_u VARCHAR(64);
DECLARE cur_1  CURSOR  FOR SELECT id,uname FROM world.t1;
OPEN cur_1;
FETCH cur_1 INTO v_i,v_u;
SELECT  v_i,v_u;
FETCH cur_1 INTO v_i,v_u;
SELECT  v_i,v_u;
FETCH cur_1 INTO v_i,v_u;
SELECT  v_i,v_u;
CLOSE cur_1;
    END$$
```

## 4.3 游标异常案例处理

**因为我们去查询表，数据行不可能是个固定值，当存储过程内的游标次数与数据行出现不一致情况下就会报错。**
**所以我们使用循环结构+游标的异常处理操作解决报错**

Bash | Copy

```
DELIMITER $$
CREATE
    PROCEDURE `world`.`p_c`()
    BEGIN
DECLARE v_i INT;                                          定义局部变量
DECLARE v_u VARCHAR(64);                                  定义局部变量
DECLARE done int default 1;                               定义游标变量（必须放在定义游标之前，否则报错）
DECLARE cur_1  CURSOR  FOR SELECT id,uname FROM world.t1; 定义游标
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 0;      定义游标异常处理

OPEN cur_1;                                               打开游标

FETCH cur_1 INTO v_i,v_u;                                 游标提取数据
while done=1                                              while循环判断异常处理值
do
SELECT  v_i,v_u;
FETCH cur_1 INTO v_i,v_u;
end while;
CLOSE cur_1;                                              关闭游标
    END$$
DELIMITER ;
```

# 5.存储过程查询及删除

Bash | Copy

```
存储过程的查询
查询整个数据存储过程的定义
select * from information_schema.ROUTINES\G
查询某个库下的存储过程
select * from information_schema.ROUTINES where routine_schema='库名'
存储过程的删除
drop procedure p_iterate;
```

%E5%AD%98%E5%82%A8%E8%BF%87%E7%A8%8B%E5%BA%94%E7%94%A8%EF%BC%89%E2%88%9A%20%7C%201.%E5%AD%98%E5%82%A8%E8%BF%87%E7%A8%8B%E5%9F%BA%E7%A1%80%