

# 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

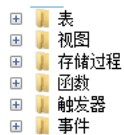
- 1.存储过程（Stored Procedure）是一种在数据库中存储复杂程序，以便外部程序调用的一种数据库对象。
- 2.存储过程是为了完成特定功能的SQL语句集，经编译创建并保存在数据库中，用户可通过指定存储过程的名字并给定参数(需要时)来调用执行。
- 3.存储过程思想上很简单，就是数据库 SQL 语言层面的代码封装与重用。

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）。

▼

Bash | Copy

```
1 DELIMITER $$          ---->          先将语句结束符号改写为$$
2 CREATE
3     [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
4     PROCEDURE `world`.`P_test`          ----> `那个库`.`创建存储过程的名`
5     proc_parameter:                      ----> 参数列表
6     [ 空 | IN | OUT | INOUT ] param_name type [参数模式]参数名 参数类型
7
8     BEGIN                          ---->          标记存储过程开始
9         过程体                      ---->          1.sql语句的叠加 2.流程控制结构 3.循环结构
10    END$$                      ---->          标记存储过程结束
11
12 DELIMITER ;                最后语句完成之后，再将语句结束符改为默认的；
13
14 语法解释：
15 DELTMITER：是用来定义语句结束标记的
16 PROCEDURE：不进行参数类型的定义（空）
17 参数模式 ：
18 IN ： 输入参数，单独传参。
19 OUT ： 输出参数，作为返回值的参数。
20 INOUT： 既输入有输出，可做输入也可做输出。
```

1.2 调用存储过程

▼

Bash | Copy

```
1 mysql> CALL 存储过程名(实参列表);
```

1.3 存储过程创建实例

1.3.1 空参数列表应用

Bash | Copy

```
1 0.创建空参数列表的存储过程
2 DELIMITER $$
3
4 CREATE
5 PROCEDURE `world`.`p_null`()      ()参数列表为空
6
7 BEGIN
8 select * from world.city where countrycode='CHN';
9 END$$
10
11 DELIMITER ;
12 2.查看创建的存储过程 (information_schema)
13      库名.表名      列名
14 mysql> select * from information_schema.routines where routine_schema='world'\G;
15 ***** 1. row *****
16      SPECIFIC_NAME: p_null
17      ROUTINE_CATALOG: def
18      ROUTINE_SCHEMA: world
19      ROUTINE_NAME: p_null
20      ROUTINE_TYPE: PROCEDURE
21      DATA_TYPE:
22 CHARACTER_MAXIMUM_LENGTH: NULL
23 CHARACTER_OCTET_LENGTH: NULL
24 NUMERIC_PRECISION: NULL
25 NUMERIC_SCALE: NULL
26 DATETIME_PRECISION: NULL
27 CHARACTER_SET_NAME: NULL
28 COLLATION_NAME: NULL
29 DTD_IDENTIFIER: NULL
30      ROUTINE_BODY: SQL
31      ROUTINE_DEFINITION: BEGIN
32 select * from world.city where countrycode='CHN';
33 END
34      EXTERNAL_NAME: NULL
35      EXTERNAL_LANGUAGE: SQL
36      PARAMETER_STYLE: SQL
37      IS_DETERMINISTIC: NO
38      SQL_DATA_ACCESS: CONTAINS SQL
39      SQL_PATH: NULL
40      SECURITY_TYPE: DEFINER
41      CREATED: 2021-04-10 20:27:11
42      LAST_ALTERED: 2021-04-10 20:27:11
43      SQL_MODE:
44      ROUTINE_COMMENT:
45      DEFINER: oldguo@10.0.0.%
46      CHARACTER_SET_CLIENT: utf8
47      COLLATION_CONNECTION: utf8_general_ci
48      DATABASE_COLLATION: utf8mb4_0900_ai_ci
49 3.调用我们创建的存储过程
50 mysql> call world.p_null();
```

51					
52	ID	Name	CountryCode	District	Population
53					
54	1890	Shanghai	CHN	Shanghai	9696300
55	1891	Peking	CHN	Peking	7472000
56	1892	Chongqing	CHN	Chongqing	6351600
57	1893	Tianjin	CHN	Tianjin	5286800
58	1894	Wuhan	CHN	Hubei	4344600
59	1895	Harbin	CHN	Heilongjiang	4289800
60	1896	Shenyang	CHN	Liaoning	4265200
61	1897	Kanton [Guangzhou]	CHN	Guangdong	4256300
62					

### 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

```

1  0.创建in参数列表的存储过程
2  DELIMITER $$
3
4  CREATE
5  PROCEDURE `world`.`p_in`(in cc char(64))  ()参数列表为in, 要设置数据类型, 比原表的更长
6
7  BEGIN
8  select * from world.city where countrycode=cc; 调用存储过程cc
9  END$$
10
11 DELIMITER ;
12
13 1. 调用存储过程, 进程传参
14 mysql> CALL p_in('CHN');
15
16 | ID | Name | CountryCode | District | Population |
17 +-----+-----+-----+-----+-----+
18 | 1890 | Shanghai | CHN | Shanghai | 9696300 |
19 | 1891 | Peking | CHN | Peking | 7472000 |
20 | 1892 | Chongqing | CHN | Chongqing | 6351600 |
21 | 1893 | Tianjin | CHN | Tianjin | 5286800 |
22 | 1894 | Wuhan | CHN | Hubei | 4344600 |
23 | 1895 | Harbin | CHN | Heilongjiang | 4289800 |
24 | 1896 | Shenyang | CHN | Liaoning | 4265200 |
25 | 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

```
1 0.创建out参数列表的存储过程
2 DELIMITER $$
3
4 CREATE
5 PROCEDURE `world`.`p_out`(out c int)    ()参数列表为out, 要设置数据类型, 比原表的更长
6
7 BEGIN
8 select count(*) from world.city into c    将sql语句查询结果赋值给c
9 select c;    因为存储过程默认不会输出, 所有需要调用c (局部变量不用@符号就可以调用)
10 END$$
11
12 DELIMITER ;
13 1.调用存储过程中输出的局部变量, 需要先在外定义用户变量。将局部变量的值赋值给用户变量, 再调用用户变量显示值。
14 1.1 定义用户变量
15 mysql> SET @a:=0;
16 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。
17 mysql> call world.p_out(@a);
18 +-----+
19 | c      |
20 +-----+
21 | 4079   |
22 +-----+
23
```

### 1.3.4 in和out 混合使用

Bash | Copy

```
1  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)
2  0.创建in out 混合参数列表的存储过程
3  DELIMITER $$
4
5  CREATE
6  PROCEDURE `world`.`p_in_out`(in cc char(64),out c char(64),out s int)  cc输入参数列表国家 c,s输出参数列表（国家id）
7
8  BEGIN
9  select countrycode,sum(population) from world.city
10 where countrycode=cc
11 group by cc
12 into c,s;
13 select c,s;      因为存储过程默认不会输出，所有需要调用c,s（局部变量不用@符号就可以调用）
14 END$$
15 DELIMITER ;
16
17 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。
18 1.1 定义用户变量
19 mysql> SET @a:=0;
20 mysql> SET @b:=0;
21 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
22 mysql> CALL world.p_in_out('CHN',@a,@b);
23 +-----+-----+
24 | c      | s      |
25 +-----+-----+
26 | CHN    | 175953614 |
27 +-----+-----+
```

### 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。

Bash | Copy

```
1  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)
2  0.创建inout参数列表的存储过程
3  DELIMITER $$
4
5  CREATE
6  PROCEDURE `world`.`p_inout`(inout cc char(64),out s int)  cc inout参数列表国家 s输出参数列表(国家总人口)
7
8  BEGIN
9  select countrycode,sum(population) from world.city
10 where countrycode=cc
11 group by cc
12 into cc,s;
13 select cc,s;      因为存储过程默认不会输出，所有需要调用cc,s（局部变量不用@符号就可以调用）
14 END$$
15 DELIMITER ;
16 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。
17 1.1 定义用户变量
18 mysql> SET @a:=0;
19 mysql> SET @b:=0;
20 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
21 mysql> CALL world.p_in_out('CHN',@a,@b);
22 +-----+-----+
23 | c      | s          |
24 +-----+-----+
25 | CHN    | 175953614  |
26 +-----+-----+
```

## 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

```

1  练习一
2
3  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
4  uname:6字符随机长度。
5
6  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。
7
8  0.首先模拟环境，创建被插入的表
9
10 mysql> use world
11
12 mysql> create table t1( id int not null primary key auto_increment,
13    uname varchar(64) not null ,
14    pass varchar(20) not null )engine=innodb charset=utf8mb4;
15
16 1.创建存储过程
17
18 DELIMITER $$
19
20 CREATE
21 PROCEDURE `world`.`p_1`()
22
23 BEGIN
24 DECLARE v_u VARCHAR(16);          声明局部变量
25 DECLARE v_p VARCHAR(64);          声明局部变量
26
27 SELECT SUBSTR('abcdefghijklmnopqrstuvwxyz',1+FLOOR(RAND()*21),6) INTO v_u;  赋值v_u
28
29 SELECT CONCAT(SUBSTR('ABCDEFGHIJKLMNOPQRSTUVWXYZ',1+FLOOR(RAND()*26),1),
30 SUBSTR(REPLACE(UUID(), '-', ''),1+FLOOR(RAND()*22),11)) INTO v_p;          赋值v_p
31
32 INSERT t1(uname,pass) VALUES(v_u,v_p);
33
34 END$$
35
36 DELIMITER ;
37
38 2.调用存储过程，对t1表进行插入一行数据
39
40 mysql> call world.p_1();
41
42 Query OK, 1 row affected (0.00 sec)
43
44
45 mysql> select * from t1;
46
47 +-----+-----+-----+
48 | id | uname | pass |
49 +-----+-----+-----+
50 | 1 | cdefgh | P1eba190000c |
51 +-----+-----+-----+

```



Bash | Copy

```

1  练习二
2  题目要求:
3  uname:随机6位字符串
4  age:18到25岁
5  gender :m或f
6  telnum 以数字1开头 第二位3-9 后面9位000000000-999999999
7  0.首先模拟环境, 创建被插入的表
8  mysql> use world
9  mysql> create table t1( id int not null primary key auto_increment,
10  uname varchar(64) not null ,
11  age tinyint not null,
12  gender char(1) not null,
13  telnum char(11) not null,
14  udate datetime not null default now() )engine=innodb charset=utf8mb4;
15  1.创建存储过程
16  DELIMITER $$
17  CREATE
18  PROCEDURE `world`.`p_2`()
19  BEGIN
20  DECLARE v_u VARCHAR(64);
21  DECLARE v_a TINYINT;
22  DECLARE v_g CHAR(1);
23  DECLARE v_t CHAR(11);
24  SELECT SUBSTR('abcdefghijklmnopqrstuvwxyz',1+FLOOR(RAND()*21),6) INTO v_u;
25  SELECT 18+FLOOR(RAND()*8) INTO v_a;
26  SELECT SUBSTR('MF',1+FLOOR(RAND()*2),1) INTO v_g;
27  SELECT CONCAT('1',3+FLOOR(RAND()*7),LPAD(FLOOR(RAND()*1000000000),9,'0')) INTO v_t;
28  INSERT INTO world.t1(uname,age,gender,telnum)
29  VALUES (v_u,v_a,v_g,v_t);
30  END$$
31  DELIMITER ;
32  2.调用存储过程
33  mysql> call world.p_2();
34  Query OK, 1 row affected (0.00 sec)
35
36  mysql> select * from t1;
37  +-----+-----+-----+-----+-----+-----+
38  | id | uname | age | gender | telnum | udate |
39  +-----+-----+-----+-----+-----+-----+
40  | 1 | efghij | 19 | M | 16398396511 | 2021-04-12 18:50:18 |
41  +-----+-----+-----+-----+-----+-----+

```

## 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

```
1  if 条件1 then 语句1;  
2  
3  elseif 条件2 then 语句2;  
4  ...;  
5  else 语句n;  
   end if;
```

### 3.2.2 CASE

```
1  方法1：等值判断  
2  case 变量|表达式|字段  
3  
4  when 判断的值 then 结果或语句  
5  when 判断的值 then 结果或语句  
6  
7  ...  
8  else 结果或语句  
9  
10 end case;  
11  
12  
13 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

```
1 练习一
2
3 题目要求：输入整数体重，判断身材形态。
4 0.创建存储过程p_if
5 DELIMITER $$
6
7 CREATE
8 PROCEDURE `world`.`p_if`(IN tz INT)
9 BEGIN
10
11     DECLARE result VARCHAR(10);
12     IF tz <100 THEN SET result='瘦';
13     ELSEIF tz BETWEEN 100 AND 130 THEN SET result='壮';
14     ELSE
15         SET result='胖';
16     END IF;
17     SELECT result;
18 END$$
19
20 DELIMITER ;
21
22 1.调用存储过程，传入参数
23 mysql> call world.p_if(120);
24
25 +-----+
26 | result |
27 +-----+
28 |  壮   |
29 +-----+
```

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

```
1  练习二
2
3  题目要求：判断年龄范围
4  0.创建存储过程
5  DELIMITER $$
6
7  CREATE
8  PROCEDURE `world`.`p_case1`(IN age INT)
9
10 BEGIN
11     DECLARE result VARCHAR(10);
12     CASE
13     WHEN age BETWEEN 0 AND 10 THEN
14     SET result='黄口小儿';
15     WHEN age BETWEEN 11 AND 20 THEN
16     SET result='青少年';
17     WHEN age BETWEEN 21 AND 30 THEN
18     SET result='青年';
19     WHEN age BETWEEN 31 AND 50 THEN
20     SET result='中年';
21     ELSE
22     SET result='退休了';
23     END CASE;
24     SELECT result;
25
26 END$$
27
28 DELIMITER ;
29
30 1.调用存储过程
mysql> CALL p_case1(23);
+-----+
| result |
+-----+
| 青年   |
+-----+
```

Bash | Copy

```
1 练习三
2
3 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。
4
5 0.模拟环境，创建用户密码表
6 mysql> create table t1(id int,username varchar(20), pass varchar(20));
7 mysql> insert into t1 values(1,'a','a'),(2,'b','b');
8 mysql> select * from t1;
9
10 +-----+-----+-----+
11 | id   | username | pass |
12 +-----+-----+-----+
13 | 1    | a       | a    |
14 | 2    | b       | b    |
15 +-----+-----+-----+
16
17 1.创建存储过程
18
19 DELIMITER $$
20
21 CREATE
22     PROCEDURE `world`.`p_case2`(IN u VARCHAR(20),IN p VARCHAR(20))
23     BEGIN
24         DECLARE result VARCHAR(20);
25         DECLARE COUNT INT DEFAULT 0;
26         SELECT COUNT(*) FROM world.t1 WHERE t1.username=u AND t1.pass=p INTO COUNT;
27         CASE WHEN COUNT>0 THEN SET result='success!';
28         ELSE
29             SET result='error!';
30         END CASE;
31         SELECT result;
32     END$$
33
34 DELIMITER ;
35
36 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

```
1. [标签: ] while 条件 do
2. 循环体;
3. end while [标签];
```

## while实例

```
1  题目：向表中插入100行数据，循环100次
2  0.模拟环境创建表
3  SELECT * FROM t1;
4  CREATE TABLE t1( id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
5  uname VARCHAR(64) NOT NULL ,
6  age TINYINT NOT NULL,
7  gender CHAR(1) NOT NULL,
8  telnum CHAR(11) NOT NULL,
9  udate DATETIME NOT NULL DEFAULT NOW() )ENGINE=INNODB CHARSET=utf8mb4;
10 1.创建存储过程
11 DELIMITER $$
12 CREATE
13     PROCEDURE `world`.`p_while`()
14     BEGIN
15         DECLARE v_u VARCHAR(64);
16         DECLARE v_a TINYINT;
17         DECLARE v_g CHAR(1);
18         DECLARE v_t CHAR(11);
19         DECLARE i INT DEFAULT 0;
20         WHILE i<100
21         DO
22             SELECT SUBSTR('abcdefghijklmnopqrstuvxyz',1+FLOOR(RAND()*21),6) INTO v_u;
23             SELECT 18+FLOOR(RAND()*8) INTO v_a;
24             SELECT SUBSTR('MF',1+FLOOR(RAND()*2),1) INTO v_g;
25             SELECT CONCAT('1',3+FLOOR(RAND()*7),LPAD(FLOOR(RAND()*1000000000),9,'0')) INTO v_t;
26             INSERT INTO world.t1(uname,age,gender,telnum)
27             VALUES (v_u,v_a,v_g,v_t);
28             SET i=i+1;
29         END WHILE;
30     END$$
31 DELIMITER ;
32 2.调用存储过程，再查看。
33 call world.p_while();
```

## 3.3.2 loop

## 介绍

死循环

## 语法

```
▼ Bash Copy
1.  [标签: ] loop
2.
3.  循环体;
    end loop [标签];
```

## loop实例+配合leave使用

Bash | Copy

```
1  题目：向表中插入100行数据，循环100次
2  0.模拟环境创建表
3  SELECT * FROM t1;
4  CREATE TABLE t1( id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
5  uname VARCHAR(64) NOT NULL ,
6  age TINYINT NOT NULL,
7  gender CHAR(1) NOT NULL,
8  telnum CHAR(11) NOT NULL,
9  udate DATETIME NOT NULL DEFAULT NOW() )ENGINE=INNODB CHARSET=utf8mb4;
10 1.创建存储过程
11 DELIMITER $$
12 CREATE
13     PROCEDURE `world`.`p_loop`()
14 BEGIN
15     DECLARE v_u VARCHAR(64);
16     DECLARE v_a TINYINT;
17     DECLARE v_g CHAR(1);
18     DECLARE v_t CHAR(11);
19     DECLARE i INT DEFAULT 1;
20     lab1:LOOP
21         SELECT SUBSTR('abcdefghijklmnopqrstuvwxy',1+FLOOR(RAND()*21),6) INTO v_u;
22         SELECT 18+FLOOR(RAND()*8) INTO v_a;
23         SELECT SUBSTR('MF',1+FLOOR(RAND()*2),1) INTO v_g;
24         SELECT CONCAT('1',3+FLOOR(RAND()*7),LPAD(FLOOR(RAND()*100000000),9,'0')) INTO v_t;
25     IF i>100
26     THEN LEAVE lab1;
27     ELSE
28         INSERT INTO world.t1(uname,age,gender,telnum)
29         VALUES (v_u,v_a,v_g,v_t);
30         SET i=i+1;
31     END IF;
32     END LOOP lab1;
33
34 END$$
35 DELIMITER ;
36 2.调用存储过程
37 CALL p_loop();
38 3.改写插入多少行存储过程
39 DELIMITER $$
40 CREATE
41     PROCEDURE `world`.`p_loop1`(IN num INT)
42 BEGIN
43     DECLARE v_u VARCHAR(64);
44     DECLARE v_a TINYINT;
45     DECLARE v_g CHAR(1);
46     DECLARE v_t CHAR(11);
47     DECLARE i INT DEFAULT 1;
48     lab1:LOOP
49         SELECT SUBSTR('abcdefghijklmnopqrstuvwxy',1+FLOOR(RAND()*21),6) INTO v_u;
50         SELECT 18+FLOOR(RAND()*8) INTO v_a;
```



```
51     SELECT SUBSTR('MF',1+FLOOR(RAND()*2),1) INTO v_g;
52     SELECT CONCAT('1',3+FLOOR(RAND()*7),LPAD(FLOOR(RAND()*1000000000),9,'0')) INTO v_t;
53     IF i>num
54     THEN LEAVE lab1;
55     ELSE
56         INSERT INTO world.t1(uname,age,gender,telnum)
57         VALUES (v_u,v_a,v_g,v_t);
58     SET i=i+1;
59     END IF;
60     END LOOP lab1;
61
62 END$$
63 DELIMITER ;
64 4.传参调用
65 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

```
1  [标签: ] repeat
2  循环体;
3  until 条件
4  end repeat [标签];
```

#### repeat实例

Bash | Copy

```
1  题目：向表中插入100行数据，循环100次
2  0.模拟环境创建表
3  SELECT * FROM t1;
4  CREATE TABLE t1( id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
5  uname VARCHAR(64) NOT NULL ,
6  age TINYINT NOT NULL,
7  gender CHAR(1) NOT NULL,
8  telnum CHAR(11) NOT NULL,
9  udate DATETIME NOT NULL DEFAULT NOW() )ENGINE=INNODB CHARSET=utf8mb4;
10 1.创建存储过程
11 DELIMITER $$
12 CREATE
13     PROCEDURE `world`.`p_repeat`()
14     BEGIN
15         DECLARE v_u VARCHAR(64);
16         DECLARE v_a TINYINT;
17         DECLARE v_g CHAR(1);
18         DECLARE v_t CHAR(11);
19         DECLARE i INT DEFAULT 0;
20         REPEAT
21             SELECT SUBSTR('abcdefghijklmnopqrstuvwxy',1+FLOOR(RAND()*21),6) INTO v_u;
22             SELECT 18+FLOOR(RAND()*8) INTO v_a;
23             SELECT SUBSTR('MF',1+FLOOR(RAND()*2),1) INTO v_g;
24             SELECT CONCAT('1',3+FLOOR(RAND()*7),LPAD(FLOOR(RAND()*100000000),9,'0')) INTO v_t;
25             INSERT INTO world.t1(uname,age,gender,telnum)
26                 VALUES (v_u,v_a,v_g,v_t);
27             SET i=i+1;
28             UNTIL i>99
29             END REPEAT;
30     END$$
31 DELIMITER ;
32 2.调用存储过程，再查看。
33 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

```
1  # 什么是游标?
2
3  保存select语句的数据集，主要用于对数据集逐行进行处理。
4  # 使用方法 需要的操作：
5  定义游标：
6
7  DECLARE cur_1 CURSOR FOR SELECT id,name FROM city;
8
9  定义游标异常处理：
10 declare done int default 1; （定义变量必须放在定义游标之前）
11 DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 0;
12
13 打开游标：
14 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

```
1 0. 模拟环境, 创建表
2 SELECT * FROM t1;
3
4 CREATE TABLE t1( id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
5 uname VARCHAR(64) NOT NULL ,
6 age TINYINT NOT NULL,
7 gender CHAR(1) NOT NULL,
8 telnum CHAR(11) NOT NULL,
9
10 update DATETIME NOT NULL DEFAULT NOW() )ENGINE=INNODB CHARSET=utf8mb4;
11
12
13
14 插入三行数据, 可以用我们之前创建的存储过程
15 mysql> call p_loop1(3);
16 mysql> SELECT * FROM t1;
17
18 +-----+-----+-----+-----+-----+-----+
19 | id | uname | age | gender | telnum | udate |
20 +-----+-----+-----+-----+-----+-----+
21 | 1 | hijklm | 18 | M | 13667132062 | 2021-04-12 22:56:31 |
22 | 2 | ghijkl | 21 | M | 17712172916 | 2021-04-12 22:56:31 |
23 | 3 | rstuvw | 18 | F | 18341998951 | 2021-04-12 22:56:31 |
24 +-----+-----+-----+-----+-----+-----+
25
26
27
28 1. 创建存储过程
29 DELIMITER $$
30 CREATE
31 PROCEDURE `world`.`p_c`()
32 BEGIN
33 DECLARE v_i INT;
34 DECLARE v_u VARCHAR(64);
35 DECLARE cur_1 CURSOR FOR SELECT id,uname FROM world.t1;
36 OPEN cur_1;
37 FETCH cur_1 INTO v_i,v_u;
38 SELECT v_i,v_u;
39 FETCH cur_1 INTO v_i,v_u;
40 SELECT v_i,v_u;
41 FETCH cur_1 INTO v_i,v_u;
42 SELECT v_i,v_u;
43 CLOSE cur_1;
44 END$$
```

## 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

```
1  DELIMITER $$
2  CREATE
3      PROCEDURE `world`.`p_c`()
4      BEGIN
5  DECLARE v_i INT;                定义局部变量
6  DECLARE v_u VARCHAR(64);        定义局部变量
7  DECLARE done int default 1;     定义游标变量（必须放在定义游标之前，否则报错）
8  DECLARE cur_1  CURSOR  FOR SELECT id,uname FROM world.t1;  定义游标
9  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 0;        定义游标异常处理
10
11  OPEN cur_1;                打开游标
12
13  FETCH cur_1 INTO v_i,v_u;    游标提取数据
14  while done=1                while循环判断异常处理值
15  do
16  SELECT  v_i,v_u;
17  FETCH cur_1 INTO v_i,v_u;
18  end while;
19  CLOSE cur_1;                关闭游标
20  END$$
21  DELIMITER ;
```

## 5.存储过程查询及删除

Bash | Copy

```
1  存储过程的查询
2  查询整个数据库存储过程的定义
3  select * from information_schema.ROUTINES\G
4  查询某个库下的存储过程
5  select * from information_schema.ROUTINES where routine_schema='库名'
6  存储过程的删除
7  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%