**存储过程之条件和异常处理**

异常处理可用在子程序中的一般流程控制。当我们希望对sql执行过程中出现的错误情况进行处理，就可以用到异常处理。如针对存储过程 、触发器或函数内部语句可能发生的错误或警告信息，需要进行相关异常或称异常的捕获，然后作出相应的处理。

**一、条件和处理程序**

**1、DECLARE条件**

　　语法：

　　DECLARE 条件名称 CONDITION FOR 条件值

条件值有如下取值：

* SQLSTATE [VALUE] sqlstate\_value
* mysql\_error\_code

sqlstate\_value参数和mysql\_error\_code参数都可以表示MySQL的错误。例如ERROR 1146 (42S02)中，sqlstate\_value值是42S02，mysql\_error\_code值是1146。这个语句指定需要特殊处理的条件。它将一个名字和指定的错误条件关联起来。这个名字可以随后被用在DECLARE HANDLER语句中。

代码：

一：使用sqlstate\_value

DECLARE can\_not\_find CONDITION FOR SQLSTATE '42S02' ;

二：使用mysql\_error\_code

DECLARE can\_not\_find CONDITION FOR **1146** ;

**2、DECLARE处理程序**

**作用域**：定义处理程序的后，接下来的begin ... End代码块，哪个错误处理定义在begin ... end内，则在该begin...end之外的错误不会被捕获。

**语法**：

**DECLARE 处理类型 HANDLER FOR 参数错误类型[,...] handler\_actions**

**处理类型为错误处理方式**，有如下的取值：

* CONTINUE表示遇到错误不处理，继续执行；
* EXIT表示遇到错误时马上退出；
* UNDO表示遇到错误后撤回之前的操作，MySQL暂不支持回滚操作；

**参数错误类型**有如下取值：

* SQLSTATE [VALUE] qlstate\_value ：这种格式是专门为ANSI SQL 和 ODBC以及其他的标准. 并不是所有的MySQL ERROR CODE 都映射到SQLSTATE。
* condition\_name ：DECLARE条件的条件名称
* SQLWARNING ：对所有以01开头的SQLSTATE代码的速记。
* NOT FOUND ：是对所有以02开头的SQLSTATE代码的速记，当然也可以代表一个游标到达数据集的末尾。
* SQLEXCEPTION ：是对所有没有被SQLWARNING或NOT FOUND捕获的SQLSTATE代码的速记。
* mysql\_error\_code:常用的mysql\_error\_code 错误码列表请参考一下网址http://dev.mysql.com/doc/refman/5.0/en/error-messages-server.html

**handler\_actions**：在执行选定的处理类型之前，需要做的操作

代码：

一：捕获sqlstate\_value

DECLARE CONTINUE HANDLER FOR SQLSTATE '42S02' SET @info='CAN NOT FIND';

二：捕获mysql\_error\_code

DECLARE CONTINUE HANDLER FOR **1146** SET @info='CAN NOT FIND';

三：先定义条件，然后调用

DECLARE can\_not\_find CONDITION FOR SQLSTATE '42S02' ;

DECLARE CONTINUE HANDLER FOR can\_not\_find SET @info='CAN NOT FIND';

四：使用SQLWARNING

DECLARE EXIT HANDLER FOR SQLWARNING SET @info='ERROR';

五：使用NOT FOUND

DECLARE EXIT HANDLER FOR NOT FOUND SET @info='CAN NOT FIND';

六：使用SQLEXCEPTION

DECLARE EXIT HANDLER FOR SQLEXCEPTION SET @info='ERROR';

**二、实例**

向表中插入重复的一条记录，由于id相同，会产生异常，此时我们可以进行异常处理。

1 -- ----------------------------

2 -- Table structure for course

3 -- ----------------------------

4 DROP TABLE IF EXISTS `course`;

5 CREATE TABLE `course` (

6 `id` int(**11**) NOT NULL,

7 `name` varchar(**255**) NOT NULL,

8 `score` int(**11**) NOT NULL,

9 PRIMARY KEY (`id`)

10 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;

11

12

13 DROP PROCEDURE IF EXISTS proc\_test\_exce;

14 CREATE PROCEDURE proc\_test\_exce(

15 IN uid INT(**11**),

16 IN uname VARCHAR(**255**),

17 IN uscore INT(**11**),

18 OUT result INT(**11**)

19 )

20 BEGIN

21 -- DECLARE EXIT HANDLER FOR SQLSTATE '23000' set result = -1;

22 DECLARE EXIT HANDLER FOR SQLWARNING,NOT FOUND,SQLEXCEPTION set result=-**1**; 23 START TRANSACTION;

24 INSERT INTO course (id, name, score) VALUES(uid, uname, uscore);

25 set result = **1**;

26 COMMIT;

27 END;

28

29 -- id 相同则返回-1

30 CALL proc\_test\_exce(**3** ,'中文', **34**, @result);

31 SELECT @result;

连续执行CALL proc\_test\_exce(3 ,'中文', 34, @result);两次，第一次reuslt返回1，第二次返回-1。因为id冲突，执行到24行，就不会再退出，不会再往下执行，而会执行22行。所以第二次返回的是-1。