**[MySQL存储过程的异常处理](https://www.cnblogs.com/geaozhang/p/6814567.html)**

*阅读目录：存储过程的异常处理*

* [*定义异常处理*](https://www.cnblogs.com/geaozhang/p/6814567.html#dingyi)
* [*单一异常处理程序*](https://www.cnblogs.com/geaozhang/p/6814567.html#dange)

*continue*

*exit*

* [*多个异常处理程序*](https://www.cnblogs.com/geaozhang/p/6814567.html#duoge)

[*关于错误编号和SQLSTATE码*](https://www.cnblogs.com/geaozhang/p/6814567.html#guanyu)

[*使用3个处理程序*](https://www.cnblogs.com/geaozhang/p/6814567.html#hulue)

[*忽略某一异常的处理*](https://www.cnblogs.com/geaozhang/p/6814567.html#hulue)

* [*异常处理的命名*](https://www.cnblogs.com/geaozhang/p/6814567.html#mingming)
* [*异常传播*](https://www.cnblogs.com/geaozhang/p/6814567.html#chuanbo)

Q：何为异常？

A：程序在执行过程中有可能出错，运行时错误叫做异常。

默认情况下，当存储过程运行出错时，过程会立即终止，并打印系统错误消息。

实验环境：

mysql> use TENNIS

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

mysql> show tables;

+-------------------+

| Tables\_in\_TENNIS |

+-------------------+

| COMMITTEE\_MEMBERS |

| MATCHES |

| PENALTIES |

| PLAYERS |

| TEAMS |

+-------------------+

5 rows in set (0.00 sec)

因为前面的实验多数用此数据库，库表结构就不再赘述了。

例：创建过程，插入一个重复的2号球队

mysql> delimiter $$

mysql> create procedure duplicate\_teams(

-> 　　out p\_processed smallint)

-> begin

-> 　　set p\_processed=1;

-> 　　insert into TEAMS values(2,27,'third');

-> 　　set p\_processed=2;

-> end $$

mysql> delimiter ;

mysql> call duplicate\_teams(@processed);

ERROR 1062 (23000): Duplicate entry '2' for key 'PRIMARY'

mysql> select @processed;

+------------+

| @processed |

+------------+

| NULL |

+------------+

解析：客户端调用存储过程，运行出错，打印错误信息，过程被终止，没有输出。

一、定义异常处理：

DECLARE ... HANDLER语句：

　　通过条件的定义和处理，可以在定义过程中，针对可能遇到的问题，做出相应的处理步骤。（也就是定义一个异常处理程序，指定当过程某条语句出错时，相应的采取什么操作）

DECLARE handler\_action HANDLER

FOR condition\_value [, condition\_value] ...

statement

handler\_action:

CONTINUE

| EXIT

condition\_value:

mysql\_error\_code

| SQLSTATE [VALUE] sqlstate\_value

| condition\_name

| SQLWARNING

| NOT FOUND

| SQLEXCEPTION

注意：declare……handler语句必须出现在变量或条件声明的后面。

　　当某个错误（condition\_value）发生时--->执行指定的语句（statement--记录错误信息），执行完之后再决定如何操作（handler\_action）。

　　1、handler\_action

　　　　continue：继续执行当前的程序(接着执行出错的SQL的下一条语句)；

　　　　exit：当前程序终止(退出当前declare所在的begin end)；

　　目前还不支持undo功能。

　　2、statement

　　　　可以是单条语句或复合语句。

　　3、condition\_value指明handler被何种条件触发；如果条件被触发，却没有handler被声明用于处理该条件，程序的进行将取决于条件类型。

二、单个异常处理程序

　　在输出参数中包含出错消息的SQLSTATE码。

例1：创建过程，插入一个重复的2号球队；continue

mysql> DELIMITER $$

mysql> CREATE PROCEDURE small\_mistake1(

-> 　　OUT error VARCHAR(5))

-> BEGIN

-> 　　DECLARE CONTINUE HANDLER FOR SQLSTATE '23000'

-> 　　SET error = '23000'; #用来记录错误发生时的一些信息

->

-> 　　select error;

-> 　　SET error = '00000';

-> 　　select error;

-> INSERT INTO TEAMS VALUES(2,27,'third'); #会出错的语句

-> SET error = '23001';

-> END$$

mysql> DELIMITER ;

mysql> call small\_mistake1(@a);

+-------+

| error |

+-------+

| NULL |

+-------+

+-------+

| error |

+-------+

| 00000 |

+-------+

mysql> select @a;

+-------+

| @a |

+-------+

| 23001 |

+-------+

实验示例解析：

　　begin end块里，定义declare……handler语句用来捕获错误(待命ing)，select、set、select顺序执行，insert语句出错，SQLSTATE码23000，捕获，进行异常处理(赋值记录)，结束后会继续执行出错的insert语句的下一条语句……

例2：创建过程，插入一个重复的2号球队；exit

mysql> DELIMITER $$

mysql> CREATE PROCEDURE small\_mistake2(

-> 　　OUT error VARCHAR(5))

-> BEGIN

-> 　　DECLARE EXIT HANDLER FOR SQLSTATE '23000'

-> 　　SET error = '23000';

->

-> 　　select error;

-> 　　SET error = '00000';

-> 　　select error;

-> 　　INSERT INTO TEAMS VALUES(2,27,'third');

-> 　　SET error = '23001';

-> END$$

mysql> DELIMITER ;

mysql> call small\_mistake2(@a);

+-------+

| error |

+-------+

| NULL |

+-------+

+-------+

| error |

+-------+

| 00000 |

+-------+

mysql> select @a;

+-------+

| @a |

+-------+

| 23000 |

+-------+

与例1唯一不同的是：handler\_action选择的是exit，表明在异常处理结束后不会继续执行错误语句后面的语句，直接退出begin end语句块。

三、多个异常处理程序

　　可以在一个过程中定义多个异常处理程序，针对不同的错误进行不同的处理。

例1：

mysql> INSERT INTO TEAMS VALUES(2,27,'third',5);

ERROR 1136 (21S01): Column count doesn't match value count at row 1

mysql> DELIMITER $$

mysql> CREATE PROCEDURE small\_mistake3(

-> 　　OUT error VARCHAR(5))

-> BEGIN

-> 　　DECLARE CONTINUE HANDLER FOR SQLSTATE '23000'

-> 　　SET error = '23000';

->

-> 　　DECLARE CONTINUE HANDLER FOR SQLSTATE '21S01'

-> 　　SET error = '21S01';

->

->　　 INSERT INTO TEAMS VALUES(2,27,'third',5); #错误语句

-> END$$

mysql> DELIMITER ;

mysql> call small\_mistake3(@error);

mysql> select @error;

+--------+

| @error |

+--------+

| 21S01 |

+--------+

解析：

　　如果SQLSTATE码是23000，异常处理执行SET error = '23000'；

　　如果SQLSTATE码是21S01，异常处理执行SET error = '21S01'；

当然，异常处理，可以是自定义，一般都是上述方式的错误信息记录。

示例3中的异常处理程序也可以使用错误编号

例如：

DECLARE CONTINUE HANDLER FOR 1062

SET error = '23000';

DECLARE CONTINUE HANDLER FOR 1136

SET error = '21S01';

关于错误编号和SQLSTATE码：

　　每个MySQL错误都有一个唯一的数字错误编号(mysql\_error\_code)，每个错误又对应一个5字符的SQLSTATE码(ANSI SQL 采用)。

SQLSTATE码对应的处理程序：

　　1、SQLWARNING处理程序：以‘01’开头的所有sqlstate码与之对应；

　　2、NOT FOUND处理程序：以‘02’开头的所有sqlstate码与之对应；

　　3、SQLEXCEPTION处理程序：不以‘01’或‘02’开头的所有sqlstate码，也就是所有未被SQLWARNING或NOT FOUND捕获的SQLSTATE(常遇到的MySQL错误就是非‘01’、‘02’开头的)

注意：‘01’、‘02’开头和‘1’、‘2’开头是有区别的，是不一样的错误sqlsate码。

*当不想为每个错误都定义一个处理程序时，可以使用3个处理程序*

*e.g：DECLARE CONTINUE HANDLER FOR SQLWARNING,NOT FOUND,SQLEXCEPTION*

mysql> DELIMITER $$

mysql> CREATE PROCEDURE small\_mistake4(

-> 　　OUT error VARCHAR(5))

-> BEGIN

-> 　　DECLARE CONTINUE HANDLER FOR SQLWARNING,NOT FOUND,SQLEXCEPTION

-> 　　SET error = 'xxxxx';

->

-> 　　INSERT INTO teams VALUES(2,27,'third');

-> END$$

mysql> DELIMITER ;

mysql> call small\_mistake4(@a);

mysql> select @a;

+-------+

| @a |

+-------+

| xxxxx |

+-------+

*如果是在定义处理程序时，想忽略某一个条件*

*e.g：DECLARE CONTINUE HANDLER FOR SQLWARNING BEGIN END;*

也就是说，当遇到SQLWARNING的问题时，进行的异常处理是begin end块，因为里面什么都没有，就类同于直接忽略。

四、异常处理的命名

　　为了提高可读性，可以给某个sqlstate代码或mysql错误代码一个名字，并且在后面的异常处理程序中使用这个名字。

DECLARE condition\_name CONDITION FOR condition\_value

condition\_value:

mysql\_error\_code

|SQLSTATE [VALUE] sqlstate\_value

1、未命名的基本格式：

BEGIN

　　DECLARE CONTINUE HANDLER FOR 1051

-- body of handler

END;

2、有命名的基本格式：

BEGIN

　　DECLARE no\_such\_table CONDITION FOR 1051;

　　DECLARE CONTINUE HANDLER FOR no\_such\_table

-- body of handler

END;

mysql> DELIMITER $$

mysql> CREATE PROCEDURE small\_mistake5(

-> 　　OUT error VARCHAR(5))

-> BEGIN

-> 　　DECLARE non\_unique CONDITION FOR SQLSTATE '23000';

-> 　　DECLARE CONTINUE HANDLER FOR non\_unique

-> 　　begin

-> 　　　　SET error = '23000';

-> 　　　　select error;

-> 　　end;

->

-> 　　INSERT INTO TEAMS VALUES(2,27,'third'); #会出错语句

-> END$$

mysql> DELIMITER ;

mysql> call small\_mistake5(@error);

+-------+

| error |

+-------+

| 23000 |

+-------+

五、异常传播

　　在嵌套块的情况下，内部块中发生异常了，首先由本块的异常处理程序来处理，如果本块没有处理，则由外部块的异常处理程序来处理。

mysql> DELIMITER $$

mysql> CREATE PROCEDURE small\_mistake6()

-> BEGIN

-> 　　DECLARE CONTINUE HANDLER FOR SQLSTATE '23000'

-> 　　SET @processed = 100;   
 ->

-> 　　BEGIN

-> 　　　　DECLARE CONTINUE HANDLER FOR SQLSTATE '21000'

-> 　　　　SET @processed = 200;

-> 　　　　INSERT INTO TEAMS VALUES(2,27,'third');

-> 　　　　set @test=123321;

-> END;

-> END$$

mysql> DELIMITER ;

mysql> call small\_mistake6;

mysql> select @processed,@test;

+------------+--------+

| @processed | @test |

+------------+--------+

| 300 | 123321 |

+------------+--------+

解析：@processed=100说明内部块里的异常传播到了外部块，交由外部块的异常处理程序进行的处理。

墙裂建议：

　　当有多层begin end的时候，每层都应该有自己完善的异常处理，做到：自己的异常，自己这层去处理。