第十章,创建和使用触发器

一,简介

MySQL的触发器和存储过程一样，都是嵌入到MySQL的一段程序。触发器是由事件来触发某个操作，这些事件包括INSERT、UPDATE和DELETE语句。如果定义了触发程序，当数据库执行了这些语句的时候就会激发触发器执行相应的操作，触发程序是与表有关的命名数据库对象，当表上出现特定事件时，将激活该对象。介绍触发器的含义、如何创建触发器、查看触发器、触发器的使用方法以及如何删除触发器。

触发器（trigger）是一个特殊的存储过程，不同的是，执行存储过程要使用CALL语句来调用，而触发器的执行不需要使用CALL语句来调用，也不需要手工启动，只要当一个预定义的事件发生的时候，就会被MySQL自动调用。比如当对fruits表进行操作（INSERT、DELETE或UPDATE）时就会激活它执行。

触发器可以查询其他表，而且可以包含复杂的SQL语句。它们主要用于满足复杂的业务规则或要求。例如，可以根据客户当前的账户状态控制是否允许插入新订单

语法格式

```
DELIMITER //
CREATE TRIGGER [触发器的名字][触发器执行时机] [触发器监测的对象]
ON [表名]
FOR EACH ROW [触发器主体代码]//
DELIMITER ;


```

说明

DELIMITER //：MySQL 默认分隔符是; 但在触发器中，我们使用 // 表示触发器的开始与结束。

[触发器的名字]：这里填写触发器的名字

[触发器执行时机]：这里设置触发器是在关键动作执行之前触发，还是执行之后触发。

[触发器监测的对象]：触发器可以监测 INSERT、UPDATE、DELETE 的操作，当监测的命令对触发器关联的表进行操作时，触发器就被激活了。

[表名]：将这个触发器与数据库中的表进行关联，触发器定义在表上，也附着在表上，如果这个表被删除了，那么这个触发器也随之被删除。

FOR EACH ROW：这句表示只要满足触发器触发条件，触发器都会被执行，也就是说带上这个参数后，触发器将监测每一行对关联表操作的代码，一旦符合条件，触发器就会被触发。

[触发器主体代码]：这里是当满足触发条件后，被触发执行的代码主体。这里可以是一句 SQL 语句，也可以是多行命令。如果是多行命令，那么这些命令要写在 BEGIN...END 之间。

二,触发器应用

记录关键字：

new 、old

触发器针对的是数据表中的每条记录（每行），每行在数据操作前后都有一个对应的状态，触发器在执行之前就将对应的状态获取到了，将没有操作之前的状态（数据）都保存到old关键字中，而操作后的状态都放到new中　　触发器针对的是数据表中的每条记录（每行），每行在数据操作前后都有一个对应的状态，触发器在执行之前就将对应的状态获取到了，将没有操作之前的状态（数据）都保存到old关键字中，而操作后的状态都放到new中

在触发器中，可以通过old和new关键字来获取绑定表中对应的记录数据

基本语法：

关键字.字段名

old和new并不是所有的触发器都有

insert：插入之前为空，没有old delete：清空数据，没有new

触发器的六种操作

BEFORE INSERT： 在插入数据前，，如不符合返回错误信息。

AFTER INSERT： 在表 A 创建新账户后，将创建成功信息自动写入表 B 中。

BEFORE UPDATE： 在更新数据前，检测更新数据是否符合业务逻辑，如不符合返回错误信息。

AFTER UPDATE： 在更新数据后，将操作行为记录在 log 中

BEFORE DELETE： 在删除数据前，检查是否有关联数据，如有，停止删除操作。

AFTER DELETE： 删除表 A 信息后，自动删除表 B 中与表 A 相关联的信息

```sql
CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));
CREATE TRIGGER ins_sum BEFORE INSERT ON account
    FOR EACH ROW SET @sum = @sum + NEW.amount;
```

创建一个account表，表中有两个字段，分别为acct_num字段（定义为int类型）和amount字段（定义成浮点类型）；其次，创建一个名为ins_sum的触发器，触发的条件是向数据表account插入数据之前，对新插入的amount字段值进行求和计算。

```
mysql> CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));
Query OK, 0 rows affected (0.07 sec)

mysql> CREATE TRIGGER ins_sum BEFORE INSERT ON account
    ->     FOR EACH ROW SET @sum = @sum + NEW.amount;
Query OK, 0 rows affected (0.02 sec)

mysql> set @sum=0;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into account values(1,1),(2,2);
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select @sum;
+------+
| @sum |
+------+
| 3.00 |
+------+
1 row in set (0.00 sec)

mysql>
```

实验条件

```sql
第一张 客户信息表（customer_information）
CREATE TABLE customer_information (
    customer_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT COMMENT '客户ID',
    customer_name VARCHAR ( 20 ) NOT NULL COMMENT '客户名称',
    customer_age TINYINT UNSIGNED NOT NULL COMMENT '客户年龄',
customer_rank VARCHAR ( 20 ) NOT NULL COMMENT '客户级别'
);
# 第二章表 客户余额表(customer_amount)
CREATE TABLE customer_amount (
```

```
    customer_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT COMMENT '客户ID',
    customer_name VARCHAR ( 20 ) NOT NULL COMMENT '客户名称',
customer_balance FLOAT NOT NULL DEFAULT 0 COMMENT '客户余额'
);
# 第三张 创建用户记录表
CREATE TABLE creation_time(
customer_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT COMMENT '客户ID',
customer_name VARCHAR ( 20 ) NOT NULL COMMENT '客户名称',
customer_status  VARCHAR ( 20 ) NOT NULL COMMENT '客户状态',
customer_date DATETIME  NOT NULL COMMENT '创建时间'
);
# 第四张 记录更新删除操作的表（new_update）
CREATE TABLE new_update (
customer_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT COMMENT '客户ID',
customer_name VARCHAR ( 20 ) NOT NULL COMMENT '客户名称',
customer_balance_old FLOAT NOT NULL DEFAULT 0 COMMENT '记录余额变化前',
customer_balance_new FLOAT NOT NULL DEFAULT 0 COMMENT '记录余额变化后',
operate_user VARCHAR(20) NOT NULL DEFAULT '记录操作用户',
update_date DATETIME  NOT NULL COMMENT '记录操作时间',
del_date varchar(20) DEFAULT "正常" COMMENT '客户状态'
);

INSERT INTO customer_information(customer_name,customer_age,customer_rank) VALU
ES('无E烦',24,'普通用户');

```

1.首先创建 BEFORE INSERT 触发器

此触发器用来记录客户的创建时间等

```
mysql> DELIMITER //
mysql> create trigger creation_found_date
    -> before insert on customer_information
    -> for each row
    -> insert into creation_time values(new.customer_id,new.customer_name,'成功',now());//
Query OK, 0 rows affected (0.09 sec)
```

2.创建一个判断用户年龄的触发器，判断用户的年龄小于100的才能创建

```
mysql> DELIMITER //
mysql> create trigger customer_age
    -> before insert on customer_information
    -> for each row
    -> if new.customer_age>100 then signal sqlstate '66666' set message_text="您的年龄有误写入失败！请填写年龄小于100岁的用户！"; end if //
Query OK, 0 rows affected (0.04 sec)
```

signal sqlstate 使用语句在存储的程序（例如存储过程，存储函数，触发器或事件）中向调用者返回错误或警告条件。语句提供了对返回值(如值和消息)的信息的控制

message_text 你要返回的内容

2.触发器 AFTER INSERT

当用户的数据插入成功以后在写入到账户余额表中

```
mysql> DELIMITER //
mysql> create trigger create_amount
    -> after insert on customer_information
    -> for each row
    -> insert into customer_amount values(new.customer_id,new.customer_name,0);//
Query OK, 0 rows affected (0.02 sec)
```

此时 无E烦网上办卡,没激活没钱 查看触发器是否启动

```
mysql> INSERT INTO customer_information(customer_name,customer_age,customer_rank) VALUES('无E烦',24,'普通用户');
Query OK, 1 row affected (0.07 sec)
```

```
mysql> SELECT * FROM customer_information;
+-------------+---------------+--------------+---------------+
| customer_id | customer_name | customer_age | customer_rank |
+-------------+---------------+--------------+---------------+
|           1 | 无E烦         |           24 | 普通用户      |
+-------------+---------------+--------------+---------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM customer_amount;
+-------------+---------------+------------------+
| customer_id | customer_name | customer_balance |
+-------------+---------------+------------------+
|           1 | 无E烦         |                0 |
+-------------+---------------+------------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM creation_time;
+-------------+---------------+-----------------+---------------------+
| customer_id | customer_name | customer_status | customer_date       |
+-------------+---------------+-----------------+---------------------+
|           1 | 无E烦         | 成功            | 2022-11-23 12:35:20 |
+-------------+---------------+-----------------+---------------------+
1 row in set (0.00 sec)
```

3.BEFORE UPDATE 在更新之前操作

这个 无E烦 用户有一天彩票中了一百万，来我们银行存钱了，那么我们银行有规矩一次性存入一百万就是我们的VIP用户，根据这个写一个触发器

```
mysql> DELIMITER //
mysql> create trigger save_money
    -> before update on customer_amount
    -> for each row if new.customer_balance>=1000000 and new.customer_name=old.customer_name then
    -> update customer_information set customer_rank='VIP' where customer_id=old.customer_id;end if//
Query OK, 0 rows affected (0.01 sec)
```

我们将存款改到数据库内

```
mysql> update customer_amount set customer_balance=1000000 where customer_name='无E烦';
Query OK, 1 row affected (0.05 sec)
```

再次查看用户有没有升级称为VIP用户

```
mysql> SELECT * FROM customer_information;
+-------------+---------------+--------------+---------------+
| customer_id | customer_name | customer_age | customer_rank |
+-------------+---------------+--------------+---------------+
|           1 | 无E烦         |           24 | VIP           |
+-------------+---------------+--------------+---------------+
1 row in set (0.01 sec)

mysql> SELECT * FROM customer_amount;
+-------------+---------------+------------------+
| customer_id | customer_name | customer_balance |
+-------------+---------------+------------------+
|           1 | 无E烦         |          1000000 |
+-------------+---------------+------------------+
1 row in set (0.00 sec)
```

4.AFTER UPDATE 触发器

用户来存钱了，然后我们修改用户余额，那么需要记录一下，出来问题可以找对应的人

无E烦又来了，这次又带来十万块钱 存入到我们银行当中

```
mysql> DELIMITER //
mysql> create trigger record_operation after update on customer_amount for each row
 insert into new_update(customer_name,customer_balance_old,customer_balance_new,ope
rate_user,update_date) values(old.customer_name,old.customer_balance,new.customer_b
alance,(SELECT USER()),now());//
Query OK, 0 rows affected (0.06 sec)
```

1. 查看更新后是否又记录

```
mysql> SELECT * FROM new_update;
Empty set (0.01 sec)

mysql>
mysql> SELECT * FROM customer_amount;
+-------------+---------------+------------------+
| customer_id | customer_name | customer_balance |
+-------------+---------------+------------------+
|           1 | 无E烦         |          1000000 |
+-------------+---------------+------------------+
1 row in set (0.00 sec)
```

5.BEFORE DELETE 触发器

删除前操作

如果有一天，有个潜入我行系统的“小波崽子”，删除可以看到我们的用户信息表，想要删除我们的客户信息中某一个人的信息，那么我们就要针对客户信息的表做一个防删除的触发器

```
mysql> create trigger del_customer
    -> before delete on customer_information
    -> for each row if old.customer_name in (select customer_name from customer_amount ) and (select customer_balanc
e from customer_amount where customer_name=old.customer_name)>0 then
    -> signal sqlstate '66666' set message_text = '这位客户还有剩余额度！无法删除！';
    -> end if //
Query OK, 0 rows affected (0.11 sec)
```

测试删除

```
mysql> delete from customer_information where customer_id=1//
ERROR 1644 (66666): 这位客户还有剩余额度！无法删除！
mysql>
```

6.AFTER DELETE触发器

删除前操作

无E烦 又来我们银行取钱了，打算全部取走并注销账户，针对这个来写一个触发器，一旦余额为0的时候就可以销户了，需要联动删除，连着客户信息一起消除，并记录操作

```
mysql> create trigger del_user
    -> after delete on customer_amount
    -> for each row
    -> if old.customer_balance=0 then
    -> delete from customer_information where customer_name=old.customer_name;
    -> update creation_time set customer_status='已销户',customer_date=NOW();
    -> insert into new_update(customer_name,customer_balance_old,customer_balance_new,operate_user,update_date,del_d
ate) values(old.customer_name,0,0,(select user()),now(),'已销户');
    -> end if//
Query OK, 0 rows affected (0.03 sec)
```

取走钱并销户

```
mysql> update customer_amount set customer_balance=0;
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> delete from customer_amount where customer_name="无E烦";
Query OK, 1 row affected (0.02 sec)
```

查看四张表

```
mysql> SELECT * FROM customer_information;
+-------------+---------------+--------------+---------------+
| customer_id | customer_name | customer_age | customer_rank |
+-------------+---------------+--------------+---------------+
|           2 | 无E烦         |           24 | 普通用户      |
+-------------+---------------+--------------+---------------+
1 row in set (0.03 sec)

mysql>
mysql> SELECT * FROM customer_amount;
+-------------+---------------+------------------+
| customer_id | customer_name | customer_balance |
+-------------+---------------+------------------+
|           2 | 无E烦         |                0 |
+-------------+---------------+------------------+
1 row in set (0.00 sec)

mysql>
mysql> SELECT * FROM creation_time;
+-------------+---------------+-----------------+---------------------+
| customer_id | customer_name | customer_status | customer_date       |
+-------------+---------------+-----------------+---------------------+
|           1 | 无E烦         | 已销户          | 2022-11-24 21:47:54 |
|           2 | 无E烦         | 成功            | 2022-11-25 11:18:25 |
+-------------+---------------+-----------------+---------------------+
2 rows in set (0.00 sec)

mysql>
mysql> SELECT * FROM new_update;
+-------------+---------------+----------------------+----------------------+---------------+---------------------+----------+
| customer_id | customer_name | customer_balance_old | customer_balance_new | operate_user  | update_date         | del_date |
+-------------+---------------+----------------------+----------------------+---------------+---------------------+----------+
|           1 | 无E烦         |              1000000 |                    0 | root@localhost | 2022-11-24 21:47:43 | 正常     |
|           2 | 无E烦         |                    0 |                    0 | root@localhost | 2022-11-24 21:47:54 | 已销户   |
+-------------+---------------+----------------------+----------------------+---------------+---------------------+----------+
2 rows in set (0.02 sec)
```