触发器（trigger）是个特殊的存储过程，它的执行不是由程序调用，也不是手工启动，而是由事件来触发，比如当对一个表进行操作（ insert，delete， update）时就会激活它执行。触发器经常用于加强数据的完整性约束和业务规则等。

触发器有一个非常好的特性就是:触发器可以禁止或回滚违反引用完整性的更改，从而取消所尝试的数据修改。

什么意思，举个例子解释一下，街机游戏大家都玩过吧，闯过一关，闯下一关，有一关没闯过就要从第一关开始。触发器根这个类似。

官方解释如下  
触发程序视为单一交易中的一部份，因此可以由原触发程序还原交易，如果在交易过程中侦测到严重的错误（如使用者中断连线），则会自动还原整个交易。  
他的作用很明显了，可以保证数据的完整性，下面有一个实例来说明他的好处，以及如果使编写代码不那么复杂

**二，触发器语法**

代码如下:

CREATE TRIGGER trigger\_name trigger\_time trigger\_event  
    ON tbl\_name FOR EACH ROW trigger\_stmt

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。

触发程序与命名为tbl\_name的表相关。tbl\_name必须引用永久性表。不能将触发程序与TEMPORARY（temporary： 临时的）表或视图关联起来。

trigger\_time是触发程序的动作时间。它可以是BEFORE或AFTER，以指明触发程序是在激活它的语句之前或之后触发。

trigger\_event指明了激活触发程序的语句的类型。trigger\_event可以是下述值之一：  
·         INSERT：将新行插入表时激活触发程序，例如，通过INSERT、LOAD DATA和REPLACE语句。  
·         UPDATE：更改某一行时激活触发程序，例如，通过UPDATE语句。  
·         DELETE：从表中删除某一行时激活触发程序，例如，通过DELETE和REPLACE语句。  
请注意，trigger\_event与以表操作方式激活触发程序的SQL语句并不很类似，这点很重要。例如，关于INSERT的BEFORE触发程序不仅能被INSERT语句激活，也能被LOAD DATA语句激活。  
可能会造成混淆的例子之一是insert into .. on duplicate update ...语法：BEFORE INSERT触发程序对于每一行将激活，后跟AFTER INSERT触发程序，或BEFORE UPDATE和AFTER UPDATE触发程序，具体情况取决于行上是否有重复键。

对于具有相同触发程序动作时间和事件的给定表，不能有两个触发程序。

例如，对于某一表，不能有两个BEFORE UPDATE触发程序。但可以有1个BEFORE UPDATE触发程序和1个BEFORE INSERT触发程序，或1个BEFORE UPDATE触发程序和1个AFTER UPDATE触发程序。

trigger\_stmt是当触发程序激活时执行的语句。如果你打算执行多个语句，可使用BEGIN ... END复合语句结构。这样，就能使用存储子程序中允许的相同语句

#产品表

CREATE TABLE Product(

proID INT AUTO\_INCREMENT NOT NULL PRIMARY KEY COMMENT '商品表主键',

price DECIMAL(**10**,**2**) NOT NULL COMMENT '商品价格',

type INT NOT NULL COMMENT '商品类别(0生鲜,1食品,2生活)',

dtime DATETIME NOT NULL COMMENT '创建时间'

)AUTO\_INCREMENT=**1** COMMENT='商品表';

商品类别汇总表

CREATE TABLE ProductType(

ID INT NOT NULL COMMENT '商品类别(0生鲜,1食品,2生活)',

amount INT NOT NULL COMMENT '每种类别商品总金额',

PRIMARY KEY (ID)

)COMMENT='商品类别资金汇总表';

产品价格变动表

CREATE TABLE Product\_log(

ID INT AUTO\_INCREMENT NOT NULL COMMENT '主键',

productid INT NOT NULL COMMENT '产品id',

newprice DECIMAL(**10**,**2**) COMMENT '更改后的价格',

oldprice DECIMAL(**10**,**2**) COMMENT '更改前的价格',

PRIMARY KEY(ID)

)AUTO\_INCREMENT=**1** COMMENT='产品价格变动表';

#插入测试数据

INSERT INTO ProductType VALUES(**1**,**0.00**),(**2**,**0.00**),(**3**,**0.00**);

## ****INSERT触发器****

insert触发器只有NEW.Column.

在Product表中建立INSERT触发器，当往Product表中插入产品时，更新ProductType表对应的分类商品价格。

delimiter $$

create trigger tr\_product\_insert after insert on product FOR EACH ROW

begin

Update producttype

set amount=amount+new.price

where id=new.type;

end $$

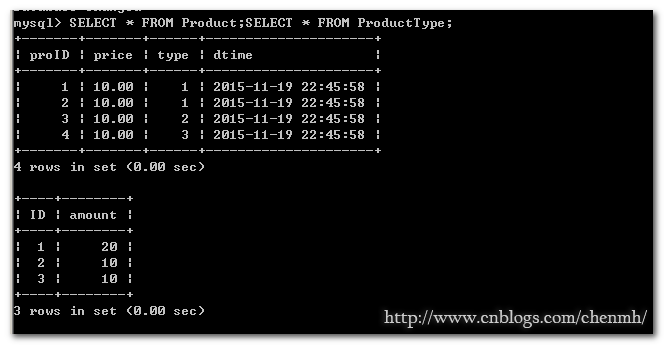
delimiter ;

**插入测试数据**

INSERT INTO Product(price,type,dtime) VALUES(**10.00**,**1**,NOW()),(**10.00**,**1**,NOW()),(**10.00**,**2**,NOW()),(**10.00**,**3**,NOW());

SELECT \* FROM Product;

SELECT \* FROM ProductType;



## **UPDATE触发器**

 update触发器中NEW.column代表更新后的值，OLD.column代表更新前的值。

**#UPDATE触发器**

**DELIMITER $$**

**create **trigger** tr\_product\_updat after update on product for each row**

**BEGIN**

**IF NEW. price<>OLD.price THEN #当价格发生变化时生成一条价格变动的日志信息插入Product\_log表 （不等于）**

**INSERT INTO Product\_log(productid,newprice,oldprice) VALUES(NEW.proID, NEW.price, OLD.price);**

**ELSE IF NEW.type<>OLD.type THEN**

**#当产品类型发生改变时更新ProductType表对应的类别**

**UPDATE ProductType**

**SET amount=amount+(SELECT price FROM Product WHERE proID=NEW.proid) WHERE ID=NEW.type;**

**UPDATE ProductType**

**SET amount=amount-(SELECT price FROM Product WHERE proID=NEW.proid) WHERE ID=OLD.type;**

**END IF;**

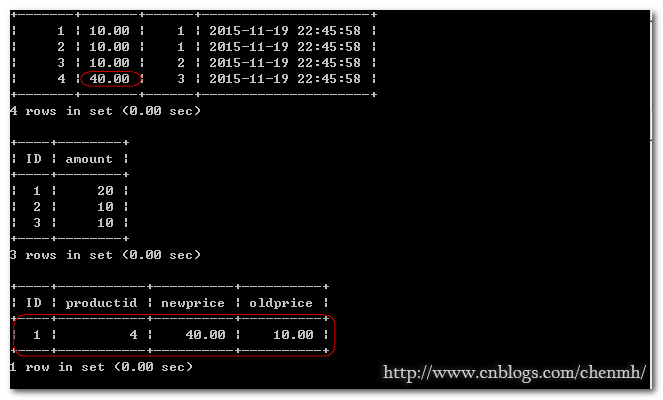
**END IF;**

**END $$**

**DELIMITER ;**

**测试数据**

UPDATE Product SET price=**40.00** WHERE proid=**4**;



UPDATE Product SET type=**2** WHERE proid=**4**;



## ****DELETE触发器****

DELIMITER $$

CREATE TRIGGER TR\_product\_delete BEFORE DELETE ON product FOR EACH row

BEGIN

UPDATE producttype

SET amount=amount-(SELECT price FROM product WHERE proID=OLD.proID)

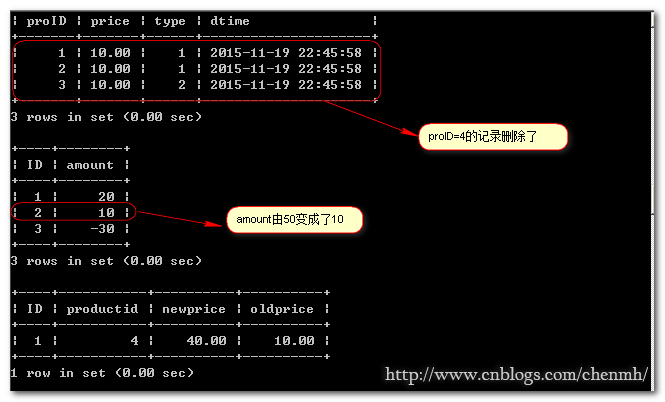
WHERE ID=OLD.type;

END $$

DELIMITER ;

**测试数据**

DELETE FROM product WHERE proID=**4**;



删除触发器的语法:

Drop trigger 触发器名

查看触发器

Show triggers

**四，触发器的优点**

1，触发器的"自动性"  
  
2，触发器的数据完整性  
触发器有回滚性，举个例子，就是你要更新五张表的数据，不会出现更新了二个张表，而另外三张表没有更新。

但是如果是用php代码去写的话，就有可能出现这种情况的，比如你更新了二张表的数据，这个时候，数据库挂掉了。有的更新了，有的没更新。这样页面显示不一致了，变有bug了。