案例：网上购物，根据生产订单的类型，商品的库存量对应的进行增和减。此案例涉及两张表，分别为订单表和商品表，下单时，商品表库存减少；退单时，商品表库存增加。如何实现？

触发器

触发器：trigger，是指事先为某张表绑定一段代码，当表中的某些内容发生改变（增、删、改）的时候，系统会自动触发代码并执行。

触发器包含三个要素，分别为

事件类型：增删改，即insert、delete和update；

触发时间：事件类型前和后，即before和after；

触发对象：表中的每一条记录（行），即整张表。

每张表只能拥有一种触发时间的一种事件类型的触发器，即每张表最多可以拥有 6 种触发器。

创建触发器

-- 创建触发器基本语法

delimiter 自定义符号 -- 临时修改语句结束符，在后续语句中只有遇到自定义符号才会结束语句

create trigger + 触发器名称 + 触发器时间 + 事件类型 on 表名 for each row

begin -- 代表触发器内容开始

-- 触发器内容主体，每行用分号结尾

end -- 代表触发器内容结束

自定义符号 -- 用于结束语句

delimiter ; -- 恢复语句结束符

根据上述案例的需求，我们先来创建两张表，分别为商品表goods和订单表orders，SQL 语句如下：

-- 创建商品表

create table goods(

id int primary key auto\_increment,

name varchar(20) not null,

price decimal(10, 2) default 0,

inventory int comment '商品库存量'

)charset utf8;

-- 插入两条数据

insert into goods values(null, 'iPhone8', 5088, 1000), (null, 'iPhoneX', 8088, 1000);

-- 创建订单表

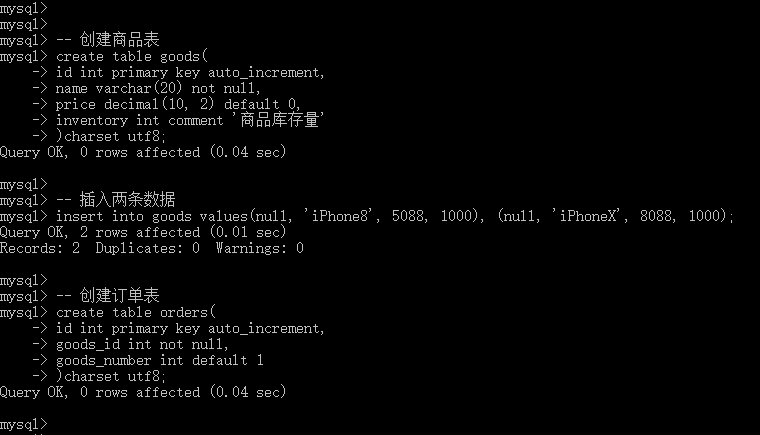
create table orders(

id int primary key auto\_increment,

goods\_id int not null,

goods\_number int default 1

)charset utf8;

[](https://camo.githubusercontent.com/6a6c143ae919f896d4cf789900e61da930eb225f/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313830313031313731343334383339)

接下来，执行如下 SQL 语句，创建触发器：

-- 创建触发器

delimiter $$ -- 临时修改语句结束符

create trigger after\_order after insert on orders for each row

begin -- 触发器内容开始

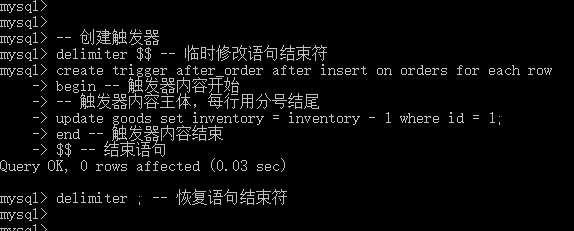
-- 触发器内容主体，每行用分号结尾

update goods set inventory = inventory - 1 where id = 1;

end -- 触发器内容结束

$$ -- 结束语句

delimiter ; -- 恢复语句结束符

[](https://camo.githubusercontent.com/abc09e49084d42fd267871aac88eca017d49ec63/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313830313031313733343133313630)

查询触发器

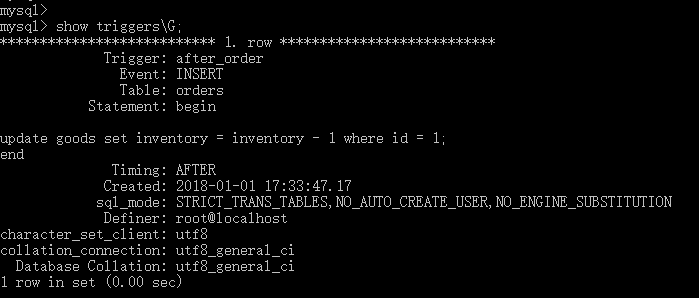
查询所有触发器或模糊匹配，

基本语法：show triggers + [like 'pattern'];

执行如下 SQL 语句，进行测试：

-- 查询所有触发器，\G 表示旋转

show triggers\G;

[](https://camo.githubusercontent.com/9299599ba8c5f18f68e385e3db5c93d006fc20bf/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313830313031313734343339353335)

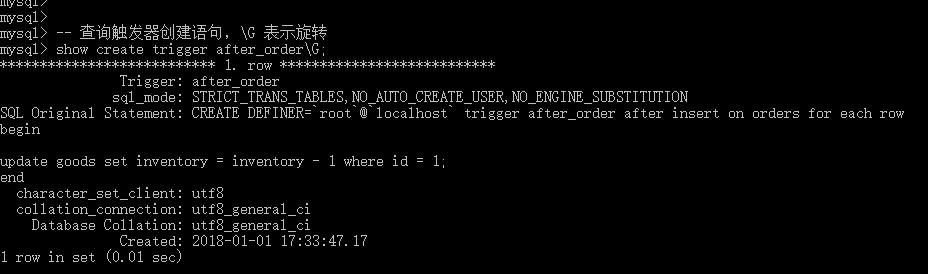
当然，我们也可以查询创建触发器的语句

基本语法：show create trigger + 触发器名称;

执行如下 SQL 语句，进行测试：

-- 查询触发器创建语句，\G 表示旋转

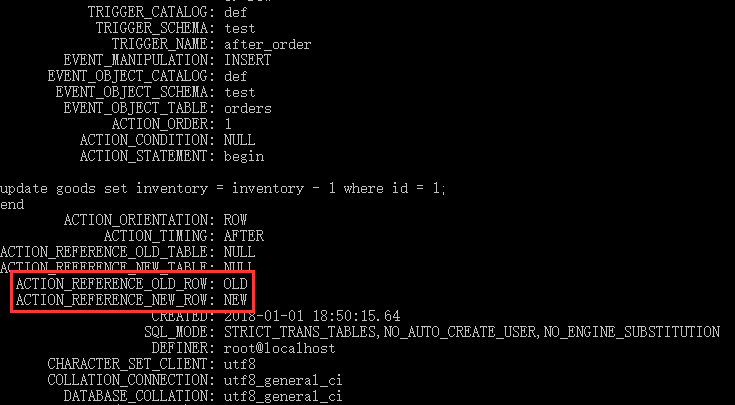
show create trigger after\_order\G;

[](https://camo.githubusercontent.com/f953e300f4b938ad818a9429aef166ab35c0cb59/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313830313031313734373039393838)

此外，所有的触发器都会被系统保持到information\_schema.triggers这张表中，执行如下 SQL，进行测试：

-- 查询触发器，\G 表示旋转

select \* from information\_schema.triggers\G;

[](https://camo.githubusercontent.com/b14e1b3f014a9f82ec87110e4b357eb21b0c24fa/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313830313031313835313438343833)

使用触发器

实际上，触发器不是我们手动触发的，而是在某种情况发生的时候自动触发，例如我们上面创建的after\_order触发器，当我们insert订单表的时候，该触发器自动执行。执行如下 SQL 语句，进行测试：

-- 查看商品表

select \* from goods;

-- 查看订单表

select \* from orders;

-- 插入订单表

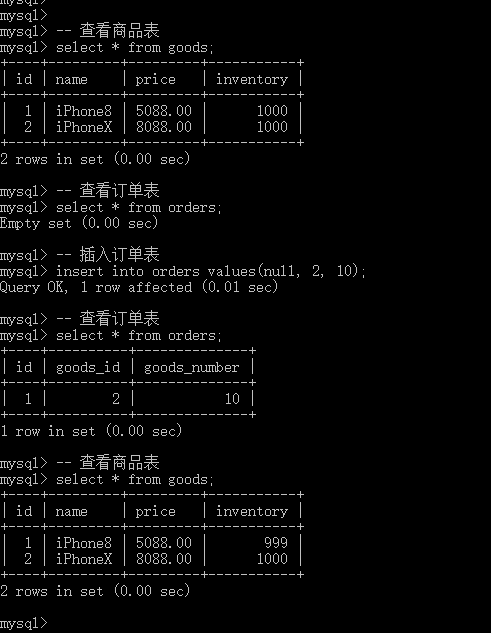
insert into orders values(null, 2, 10);

-- 查看订单表

select \* from orders;

-- 查看商品表

select \* from goods;

[](https://camo.githubusercontent.com/c700b6cd81c2d8d02704e9e5ffe28d217fa9b5c0/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313830313031313831303334313532)

观察上图，我们会发现：触发器确实生效了，在我们向orders表insert数据的时候，goods表发生了变化；但是其并没有如我们期望那样执行，就算我们将goods\_id设置为2，goods\_number设置为10，触发器操作的仍然是goods表中id为 1的记录且库存量只减1。且先不提这个问题，在创建触发器的时候，我们要特别注意：触发器的触发对象和事件类型，决不能同触发器主体的内容相同，防止发生死循环。

修改触发器 & 删除触发器

触发器不能修改，只能删除。因此，当我们需要修改触发器的时候，唯一的方法就是：先删除，后新增。

基本语法：drop trigger + 触发器名称;

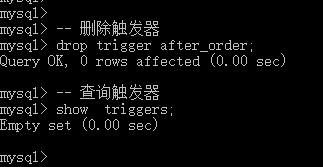
执行如下 SQL 语句，进行测试：

-- 删除触发器

drop trigger after\_order;

-- 查询触发器

show triggers;

[](https://camo.githubusercontent.com/08ff4d0377d09ff3d84ed3b5a51e0f319d752e2d/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313830313031313832383236363435)

触发器记录

触发器记录：无论触发器是否触发，只要当某种操作准备执行，系统就会将当前操作的记录的当前状态和即将执行之后的状态分别记录下来，供触发器使用。其中，当前状态被保存到old中，操作之后的状态被保存到new中。至于old和new是什么鬼？不知道大家是否还记得查看表information\_schema.triggers的时候，标红的两个字段：

ACTION\_REFERENCE\_OLD\_ROW：OLD

ACTION\_REFERENCE\_NEW\_ROW：NEW

其中，

OLD，代表是旧记录，也就是当前记录的状态，插入时没有OLD；

NEW，代表是新记录，也就是假设操作发生之后记录的状态，删除时没有NEW。

无论OLD还是 NEW，都代表记录本身，而且任何一条记录除了有数据，还有字段名。因此，使用OLD和 NEW的方法就是：

基本语法：OLD/NEW + . + 字段名

在这里，我们就能够通过触发器记录解决刚才after\_order触发器的问题了。依次执行如下 SQL 语句，进行测试：

-- 创建新触发器

delimiter $$ -- 临时修改语句结束符

create trigger after\_order\_new after insert on orders for each row

begin -- 触发器内容开始

-- 触发器内容主体，每行用分号结尾

update goods set inventory = inventory - NEW.goods\_number where id = NEW.goods\_id;

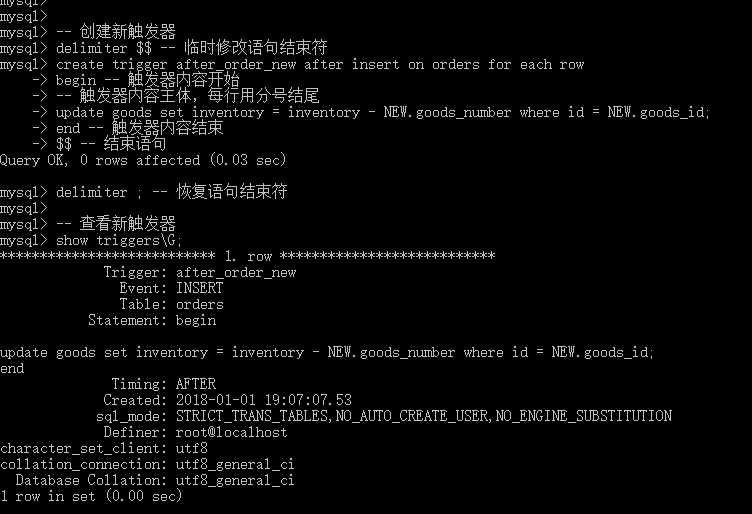
end -- 触发器内容结束

$$ -- 结束语句

delimiter ; -- 恢复语句结束符

-- 查看新触发器

show triggers\G;

[](https://camo.githubusercontent.com/30fc292a72f917885b591894ec2559921192cb2a/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313830313031313930373339303931)

-- 查看商品表

select \* from goods;

-- 查看订单表

select \* from orders;

-- 插入订单表

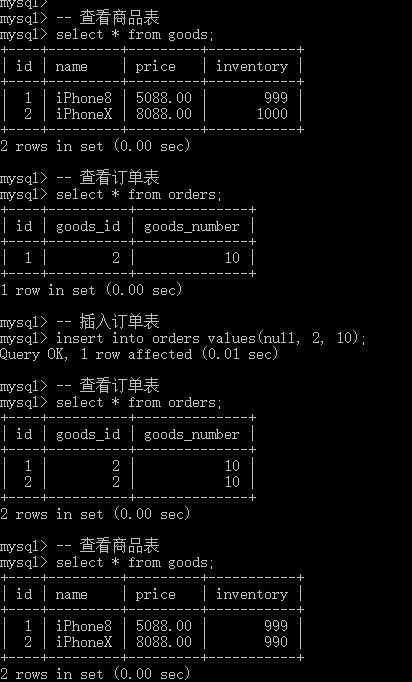
insert into orders values(null, 2, 10);

-- 查看订单表

select \* from orders;

-- 查看商品表

select \* from goods;

[](https://camo.githubusercontent.com/eb39666d55ee91f59782a393f4b0e751fa1c1310/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313830313031313930383535353732)

如上图所示，显然after\_order\_new触发器按我们预期那样正确的工作啦！