2015-2016 年度

中山大学本科生实验报告

科目：数据库实验　　**教师**：郑贵锋

年级	13 级	专业（方向）	移动信息工程
学号	13354485	姓名	朱琳
电话	13726231932	Email	280273861@qq.com

一 实验目的

学习触发器和视图

二 实验内容

I.触发器

小明是一个很用功的孩子，在毕业后他找到了一份在以电商为主业务的 BaliAA 公司的实习。火热的双 11 如期而至，主管放心不过小明的能力，便让小明负责简单的数据库维护模块，工作内容如下：

建立如下的表，名为 goods，

id	name	price	number
1001	T-Shirt	20	1000
1002	Mobile-pho	1000	20
1003	Cup	10	800

以及 orders：该表初始时为空，当有订单时插入数据，order_id 从 1401 开始自增长。

Order_id	Good_id	number

要求满足以下条件的触发器：

1.每次下单，库存都会相应的减少。

2.T-shirt 每次限购 20 件。

3.当订单被取消以后，库存会相应增加。

4.修改订单时，库存也会有相应变化。

定义完成以后，完成下列问题并截图：

1.小明的基友小绿下单，购买了 100 件 T-shirt，显示操作语句以及两个表的结果。

2.小明的女神小红下单，购买了 10 部手机，显示操作语句以及两个表的结果。

3.小红发现自己没有那么多的肾来买手机，遂请小明取消 2 中的订单，显示操作语句以及两个表的结果。

4.小绿发现是自己手抖，多按了一个 0，遂请小明修改 1 的订单，改为购买 10 件 T-shirt，显示操作语句以及两个表的结果。

II.视图

双 11 过去了，主管想请小明展示一下双 11 的成果。

1.从公司的角度，想看到每次交易的细节，请创建视图，展示出订单号、货物号、货物名称以及购买数目。

2.从商家的角度，创建视图展示出订单号、货物号、货物名称以及交易金额。

三 实验结果

1. 定义

```
DROP TABLE IF EXISTS goods;
DROP TABLE IF EXISTS orders;

CREATE TABLE goods
(
    id int primary key not null,
    name varchar(20),
    price int,
    number int
);

CREATE TABLE orders
(
    order_id int primary key not null AUTO_INCREMENT,
    good_id int,
    number int
)AUTO_INCREMENT=1401;

insert into goods values(1001,"T-Shirt",20,1000);
insert into goods values(1002,"Mobile_pho",1000,20);
insert into goods values(1003,"Cup",10,800);
```

```
delimiter $
create trigger before_make_order
before insert on orders
for each row
begin
    if new.number>20 and new.good_id=1001 then
        set new.number=20;
    end if;
end $
```

```
create trigger make_order
after insert on orders
for each row
begin
    update goods set number=goods.number-new.number where goods.id=new.good_id;
end $
```

```
create trigger delete_order
after delete on orders
for each row
begin
    update goods set number=goods.number+old.number where goods.id=old.good_id;
end $
```

```
create trigger update_order
after update on orders
for each row
begin
    update goods set number=goods.number+old.number-new.number where goods.id=old.good_id;
end $
delimiter ;
```

【分析】before_make_order 是在插入之前（即下订单之前），检查是否买的货物是 T-shirt，如果是，并且下单的数量大于 20 件，那么就强制改为 20 件。

make_order 是在下单之后，更新 goods 表的库存信息。

Delete_order 是在取消订单之后，更新 goods 表的库存信息。

Update_order 触发器是在更新订单信息之后，更新 goods 表的库存信息。

表格建立后导入数据库：

```
Database changed
mysql> select * from goods;
+------+------------+-------+--------+
| id   | name       | price | number |
+------+------------+-------+--------+
| 1001 | T-Shirt    |    20 |   1000 |
| 1002 | Mobile_pho |  1000 |     20 |
| 1003 | Cup        |    10 |    800 |
+------+------------+-------+--------+
3 rows in set (0.13 sec)
```

2. 问题回答

（1）小明的基友小绿下单，购买了 100 件 T-shirt，显示操作语句以及两个表的结果。

```
mysql> insert into orders(good_id,number) values(1001,100);
Query OK, 1 row affected (0.32 sec)

mysql> select * from orders;
+----------+---------+--------+
| order_id | good_id | number |
+----------+---------+--------+
|     1401 |    1001 |     20 |
+----------+---------+--------+
1 row in set (0.00 sec)

mysql> select * from goods;
+------+------------+-------+--------+
| id   | name       | price | number |
+------+------------+-------+--------+
| 1001 | T-Shirt    |    20 |    980 |
| 1002 | Mobile_pho |  1000 |     20 |
| 1003 | Cup        |    10 |    800 |
+------+------------+-------+--------+
3 rows in set (0.00 sec)
```

【分析】发现小绿下单的 T-shirt 数量大于 20，然后数据库直接强制将订单的数量改成了 20，并且 T-shirt 的库存由原来的 1000 件变成了 980 件，即减少了 20 件。符合预期。

（2）小明的女神小红下单，购买了 10 部手机，显示操作语句以及两个表的结果。

```
mysql> insert into orders(good_id,number) values(1002,10);
Query OK, 1 row affected (0.12 sec)

mysql> select * from orders;
+----------+---------+--------+
| order_id | good_id | number |
+----------+---------+--------+
|     1401 |    1001 |     20 |
|     1402 |    1002 |     10 |
+----------+---------+--------+
2 rows in set (0.00 sec)

mysql> select * from goods;
+------+------------+-------+--------+
| id   | name       | price | number |
+------+------------+-------+--------+
| 1001 | T-Shirt    |    20 |    980 |
| 1002 | Mobile_pho |  1000 |     10 |
| 1003 | Cup        |    10 |    800 |
+------+------------+-------+--------+
3 rows in set (0.00 sec)
```

【分析】可以看到我并没有插入订单编号，订单编号是自增到（1402），下单信息正常，和手机的库存从 20 变成了 10，库存信息更新正常。

（3）小红发现自己没有那么多的肾来买手机，遂请小明取消 2 中的订单，显示操作语句以及两个表的结果。

```
mysql> delete from orders where order_id=1402;
Query OK, 1 row affected (0.12 sec)

mysql> select * from orders;
+----------+---------+--------+
| order_id | good_id | number |
+----------+---------+--------+
|     1401 |    1001 |     20 |
+----------+---------+--------+
1 row in set (0.00 sec)

mysql> select * from goods;
+------+------------+-------+--------+
| id   | name       | price | number |
+------+------------+-------+--------+
| 1001 | T-Shirt    |    20 |    980 |
| 1002 | Mobile_pho |  1000 |     20 |
| 1003 | Cup        |    10 |    800 |
+------+------------+-------+--------+
3 rows in set (0.00 sec)
```

【分析】可以看到，当删除了订单 1402 的信息之后，orders 表中没有了原来的手机的订单信息，然后 goods 中手机的库存又变回了 20。

（4）小绿发现是自己手抖，多按了一个 0，遂请小明修改 1 的订单，改为购买 10 件 T-shirt，显示操作语句以及两个表的结果。

```
mysql> update orders set number=10 where order_id=1401;
Query OK, 1 row affected (0.08 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from orders;
+----------+---------+--------+
| order_id | good_id | number |
+----------+---------+--------+
|     1401 |    1001 |     10 |
+----------+---------+--------+
1 row in set (0.00 sec)

mysql> select * from goods;
+------+------------+-------+--------+
| id   | name       | price | number |
+------+------------+-------+--------+
| 1001 | T-Shirt    |    20 |    990 |
| 1002 | Mobile_pho |  1000 |     20 |
| 1003 | Cup        |    10 |    800 |
+------+------------+-------+--------+
3 rows in set (0.00 sec)
```

【分析】如图所示，当我更新订单信息后，orders 表中 number 的数量由 20 改为 10，同时 goods 表中库存的数量相应的比之前增加了 20-10=10。

3.视图

（1）从公司的角度，想看到每次交易的细节，请创建视图，展示出订单号、货物号、货物名称以及购买数目。

```
mysql> create view company_view AS
    -> select O.order_id,O.good_id,G.name,O.number
    -> from orders O,goods G
    -> where O.good_id=G.id;
Query OK, 0 rows affected (0.14 sec)

mysql> select * from company_view;
+----------+---------+---------+--------+
| order_id | good_id | name    | number |
+----------+---------+---------+--------+
|     1401 |    1001 | T-Shirt |     10 |
+----------+---------+---------+--------+
1 row in set (0.00 sec)
```

（2）从商家的角度，创建视图展示出订单号、货物号、货物名称以及交易金

```
mysql> create view merchant_view(order_id,good_id,name,whole_price) AS
    -> select O.order_id,O.good_id,G.name,O.number*G.price
    -> from orders O,goods G
    -> where O.good_id=G.id;
Query OK, 0 rows affected (0.03 sec)

mysql> select * from merchant_view;
+----------+---------+---------+-------------+
| order_id | good_id | name    | whole_price |
+----------+---------+---------+-------------+
|     1401 |    1001 | T-Shirt |         200 |
+----------+---------+---------+-------------+
1 row in set (0.00 sec)
```

【分析】建立如上视图，并修改视图每列的名称。交易金在原来的两张表中并没有出现，因此我们需要计算交易数量与商品单价的乘积。

四 实验感想

1.关于 mysql 中 ifelse 的使用还是第一次接触，大概就是

if [条件] then

begin

[执行语句]

end

Elseif [条件] then

Begin

[执行语句]

End

Else...

End if;

如此，同样若执行语句只有一句，可以省略 begin 和 end。

2.最初我想在订单插入之前写一个触发器检查 T-shirt 的购买数量是否合法，然而我想当然的这样写：

```
if new.number>20 and new.good_id=1001 then
    new.number=20;
end if;
```

然后报错。于是想尝试别的方式，但是若改成 update 又是不可以的，毕竟这个信息还没有插入。那么或许可以改成在插入之后再将 number 的值 update 成 20，总觉得有些违和。询问 TA 得知要写 set new.number=20 就完美解决了。

3.关于自增变量的初始化：

```
CREATE TABLE orders
(
    order_id int primary key not null AUTO_INCREMENT,
    good_id int,
    number int
)AUTO_INCREMENT=1401;
```