**数据库实验报告**

**实 验 人： 凌国明 学 号： 21307077 日 期： 2023/11/25**

**院（系）： 计算机学院 专业（班级）： 计算机科学与技术**

**联系方式：**

**实验题目： 实验4：触发器实验**

* + - 1. **实验目的**

掌握数据库触发器的设计和使用方法。

* + - 1. **实验内容和要求**

定义BEFORE触发器和AFTER触发器。能够理解不同类型触发器的作用和执行原理，验证触发器的有效性。

* + - 1. **实验重点和难点**

重点：触发器的定义

难点：利用触发器实现较为复杂的用户自定义完整性。

* + - 1. **实验工具**

MySQL、SQL Server、Navicat

* + - 1. **实验过程**
* 在MySQL环境下，执行以下操作
  + 参阅1(触发器)：<https://blog.csdn.net/little__SuperMan/article/details/123519033>
  + 参阅2(异常处理)：
    - <http://www.yiidian.com/mysql/mysql-signal-resignal.html>
    - <https://ziruchu.com/art/198>

(1) AFTER触发器

①在Lineitem表上定义一个UPDATE触发器，当修改订单明细(即修改订单明细价格extendedprice、折扣discount、税率tax) 时， 自动修改订单Orders的TotalPrice， 以保持数据一致性。

/\* totalprice = totalprice + extendedprice \* (1-discount) \*(1+tax) ) \*/

CREATE TRIGGER TRI\_Lineitem\_Price\_UPDATE

AFTER UPDATE ON Lineitem

FOR EACH ROW

BEGIN

DECLARE L\_valuediff REAL;

/\*订单明细修改后，计算订单含税折扣价总价的修正值\*/

set L\_valuediff = NEW.extendedprice\*(1-NEW.discount) \*(1+NEW.tax)

- OLD.extendedprice\*(1-OLD.discount) \*(1+OLD.tax) ;

/\*更新订单的含税折扣价总价\*/

UPDATE Orders

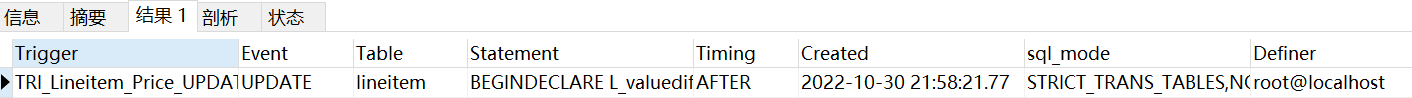
SET totalprice=totalprice+L\_valuediff

WHERE orderkey=NEW.orderkey;

END;

验证截图如下：

show TRIGGERS;



或者



②在Lineitem表上定义一个INSERT触发器， 当增加一项订单明细时， 自动修改订单Orders的TotalPrice， 以保持数据一致性。

CREATE TRIGGER TRI\_Lineitem\_Price\_INSERT

AFTER INSERT ON Lineitem

FOR EACH ROW

BEGIN

DECLARE L\_valuediff REAL;

SET L\_valuediff=NEW.extendedprice\*(1-NEW.discount) \*(1+NEW.tax) ;

/\*增加订单明细项后，计算订单含税折扣价总价的修正值\*/

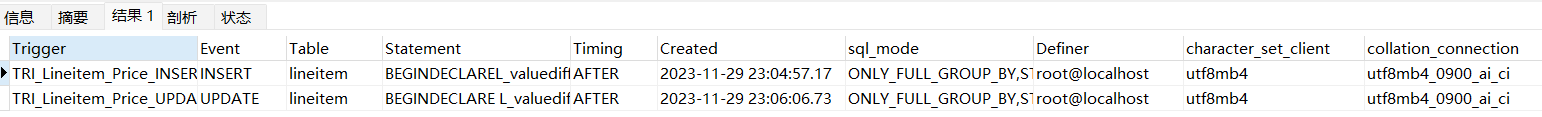
UPDATE Orders SET TotalPrice=TotalPrice+L\_valuediff

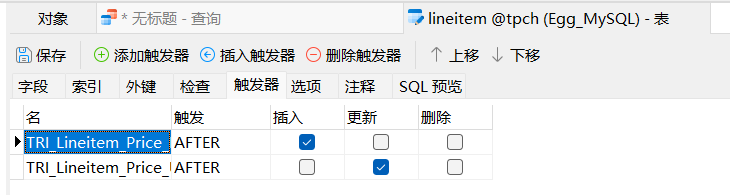
/\*更新订单的含税折扣价总价\*/

WHERE orderkey=NEW.orderkey;

END;

验证截图如下：





Insert触发器用于：当增加一项订单明细时， 自动修改订单Orders的TotalPrice

③在Lineitem表上定义一个DELETE触发器， 当删除一项订单明细时， 自动修改订单Orders的TotalPrice， 以保持数据一致性。

CREATE TRIGGER TRI\_Lineitem\_Price\_DELETE

AFTER DELETE ON Lineitem

FOR EACH ROW

BEGIN

DECLARE L\_valuediff REAL;

Set L\_valuediff = -OLD.extendedprice\*(1-OLD.discount) \*(1+OLD.tax) ;

/\*删除订单明细项后，计算订单含税折扣价总价的修正值\*/

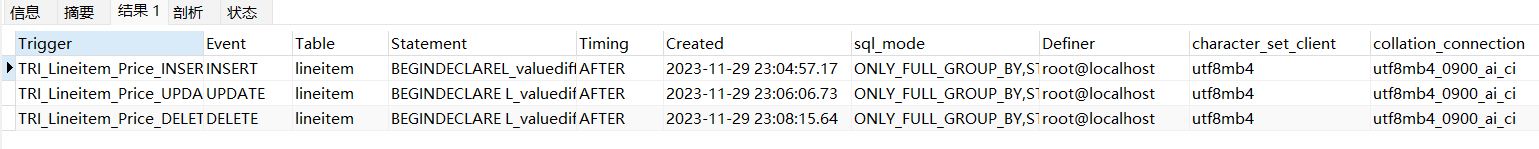
UPDATE Orders SET TotalPrice=TotalPrice+L\_valuediff

/\*更新订单的含税折扣价总价\*/

WHERE orderkey=OLD.orderkey;

END;

验证截图如下：





④验证触发器TRI\_Lineitem\_Price\_UPDATE。

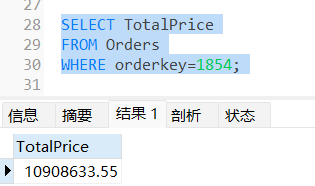
/\*查看1854号订单的含税折扣总价TotalPrice\*/

SELECT TotalPrice

FROM Orders

WHERE orderkey=1854;

验证截图如下：



/\*激活触发器：修改1854号订单第一个明细项的税率，该税率增加0.5%\*/

UPDATE Lineitem SET tax=tax+0.005

WHERE orderkey=1854 AND linenumber=1;

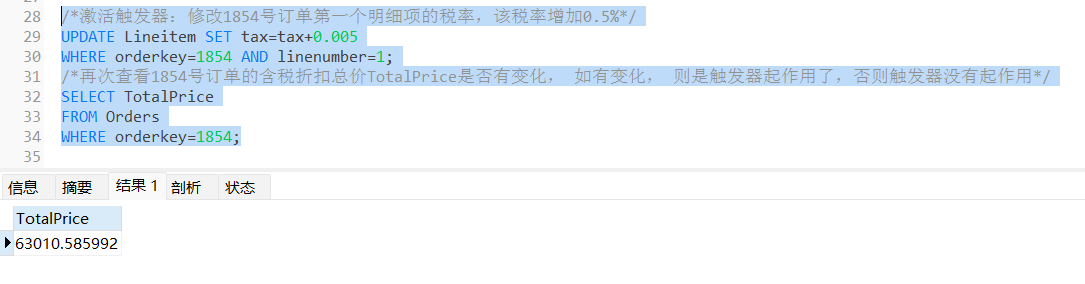
/\*再次查看1854号订单的含税折扣总价TotalPrice是否有变化， 如有变化， 则是触发器起作用了，否则触发器没有起作用\*/

SELECT TotalPrice

FROM Orders

WHERE orderkey=1854;

验证截图如下：



(2) BEFORE触发器

①在Lineitem表上定义一个BEFORE UPDATE触发器， 当修改订单明细中的数量(quantity) 时， 先检查供应表PartSupp中的可用数量availqty是否足够。

CREATE TRIGGER TRI\_Lineitem\_Quantity\_UPDATE

BEFORE UPDATE ON Lineitem

FOR EACH ROW

BEGIN

DECLARE L\_valuediff INTEGER;

DECLARE L\_availqty INTEGER;

/\*计算订单明细项修改时，订购数量的变化值\*/

set L\_valuediff=NEW.quantity-OLD.quantity;

/\*查询当前订单明细项对应零件供应记录中的可用数量\*/

SELECT availqty INTO L\_availqty

FROM PartSupp

WHERE partkey=NEW.partkey AND suppkey=NEW.suppkey;

IF(L\_availqty-L\_valuediff>=0) THEN

BEGIN

/\*如果可用数量可以满足订单订购数量， 则提示ENOUGH\*/

SELECT 'Available quantity is ENOUGH' into @user\_prompt;

/\*修改当前订单明细项对应零件供应记录中的可用数量\*/

UPDATE PartSupp

SET availqty=availqty-L\_valuediff

WHERE partkey=NEW.partkey AND suppkey=NEW.suppkey;

END;

ELSE

/\*如果可用数量不能满足订单订购数量，则更新过程异常中断\*/

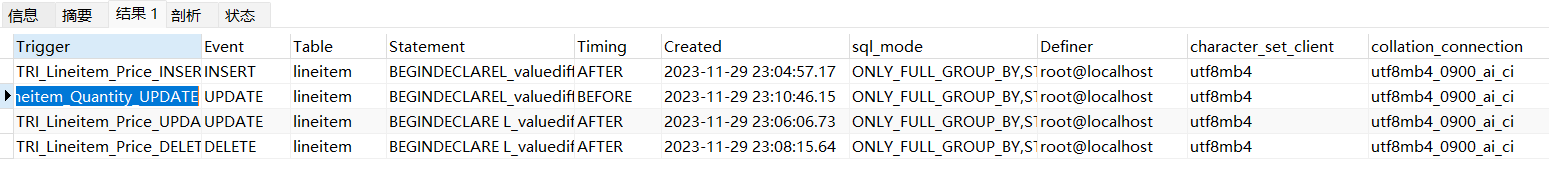
SIGNAL SQLSTATE '45000'

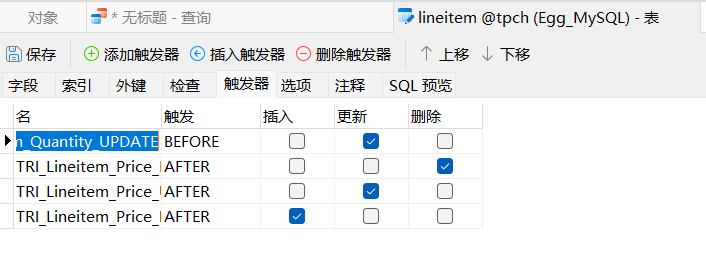
SET MESSAGE\_TEXT = 'Available quantity is NOT ENOUGH';

END IF;

END;

验证截图如下：





②在Lineitem表上定义一个BEFORE INSERT触发器， 当插入订单明细， 先检查供应表PartSupp中的可用数量availqty是否足够。

CREATE TRIGGER TRI\_Lineitem\_Quantity\_INSERT

BEFORE INSERT ON Lineitem

FOR EACH ROW

BEGIN

DECLARE L\_valuediff, L\_availqty INTEGER;

set L\_valuediff = NEW.quantity;/\*获得插入订单明细项的订购数量\*/

/\*查询当前订单明细项对应零件供应记录中的可用数量\*/

SELECT availqty INTO L\_availqty

FROM PartSupp

WHERE partkey=NEW.partkey AND suppkey=NEW.suppkey;

IF(L\_availqty - L\_valuediff>=0) THEN

BEGIN

/\*如果可用数量可以满足订单订购数量， 则提示ENOUGH\*/

SELECT 'Available quantity is ENOUGH' into @user\_prompt;

/\*修改当前订单明细项对应零件供应记录中的可用数量\*/

UPDATE PartSupp

SET availqty = availqty-L\_valuediff

WHERE partkey=NEW.partkey AND suppkey=NEW.suppkey;

END;

ELSE

/\*如果可用数量不能满足订单订购数量，则插入过程异常中断。\*/

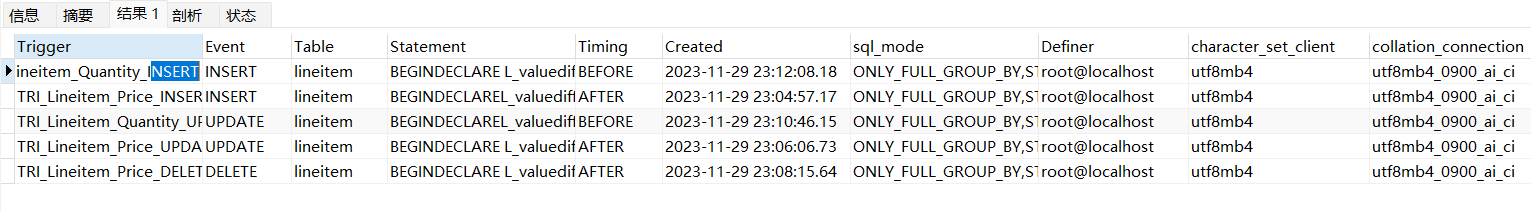
SIGNAL SQLSTATE '45000'

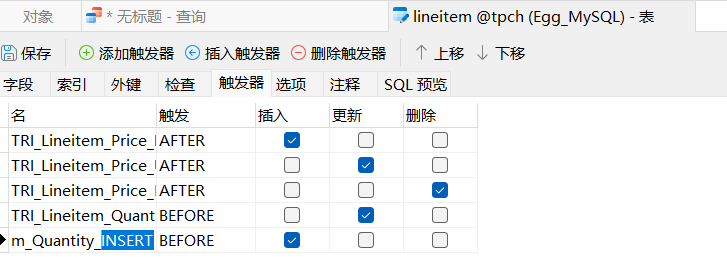
SET MESSAGE\_TEXT = 'Available quantity is NOT ENOUGH';

END IF;

END;

验证截图如下：





③在Lineitem表上定义一个BEFORE DELETE触发器， 当删除订单明细时， 该订单明细项订购的数量要归还对应的零件供应记录。

CREATE TRIGGER TRI\_Lineitem\_Quantity\_DELETE

BEFORE DELETE ON Lineitem

FOR EACH ROW

BEGIN

DECLARE L\_valuediff, L\_availqty INTEGER;

/\*获得删除订单明细项的订购数量\*/

set L\_valuediff = OLD.quantity;

/\*修改当前订单明细项对应零件供应记录中的可用数量量\*/

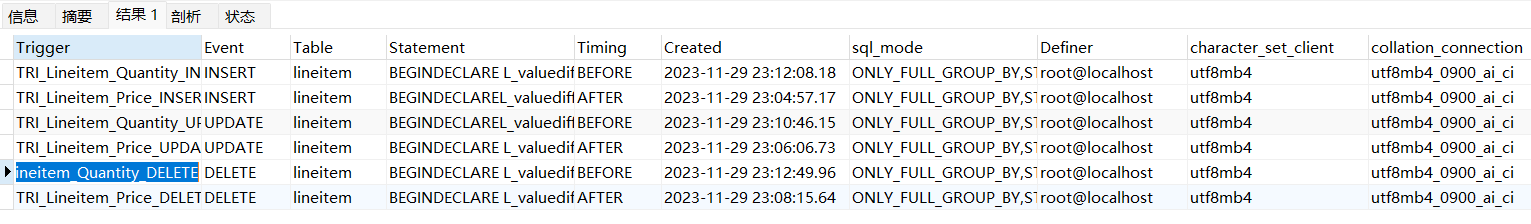
UPDATE PartSupp

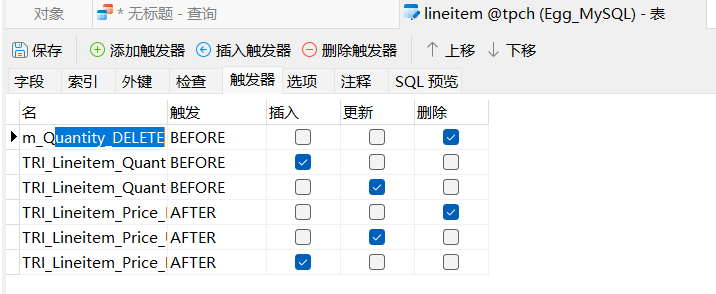
SET availqty = availqty+L\_valuediff

WHERE partkey=OLD.partkey AND suppkey=OLD.suppkey;

END;

验证截图如下：





④验证触发器TRI\_Lineitem\_Quantity\_UPDATE。

* 验证1

/\*查看1854号订单第1个明细项的零件和供应商编号、订购数量、可用数量\*/

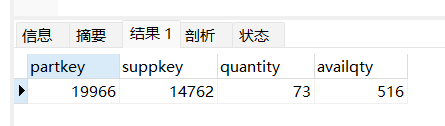
SELECT L.partkey, L.suppkey, L.quantity, PS.availqty

FROM Lineitem L, PartSupp PS

WHERE L.partkey=PS.partkey AND L.suppkey=PS.suppkey

AND L.orderkey=1854 AND L.linenumber=1;

验证截图如下：



/\*激活触发器：修改1854号订单第1个明细项的订购数量\*/

UPDATE Lineitem

SET quantity =quantity+ 5

WHERE orderkey= 1854 AND linenumber=1;

/\*再次查看1854号订单第1个明细项的相关信息,以验证触发器是否起作用\*/

SELECT L.partkey, L.suppkey, L.quantity, PS.availqty

FROM Lineitem L, PartSupp PS

WHERE L.partkey=PS.partkey AND L.suppkey=PS.suppkey

AND L.orderkey=1854 AND L.linenumber=1;

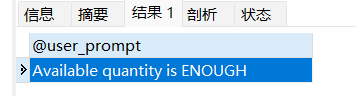
验证截图如下：



* 验证2：可用数量足够的提示？

SELECT @user\_prompt;

验证截图如下：



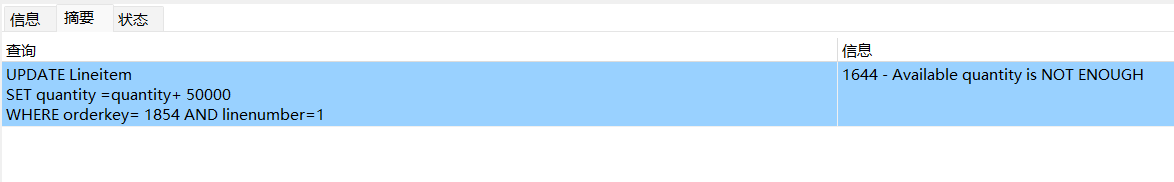
* 验证3：可用数量不够的提示？

UPDATE Lineitem

SET quantity =quantity+ 50000

WHERE orderkey= 1854 AND linenumber=1;

验证截图如下：

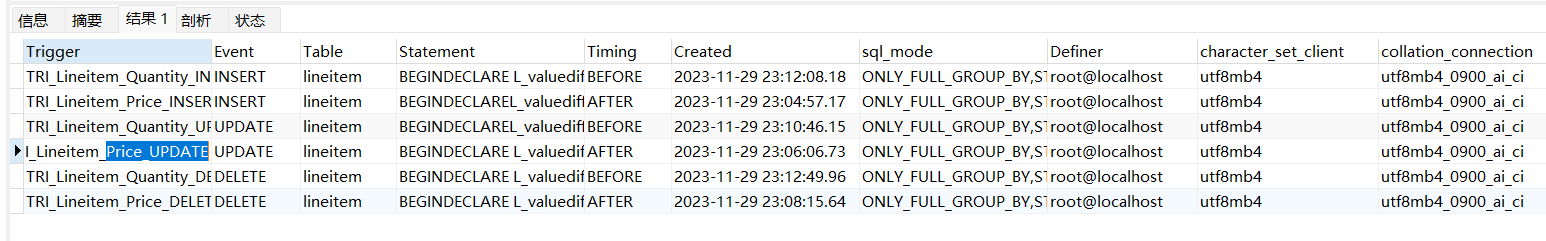


(3)删除触发器

删除触发器TRI\_Lineitem\_Price\_UPDATE。

SHOW TRIGGERS

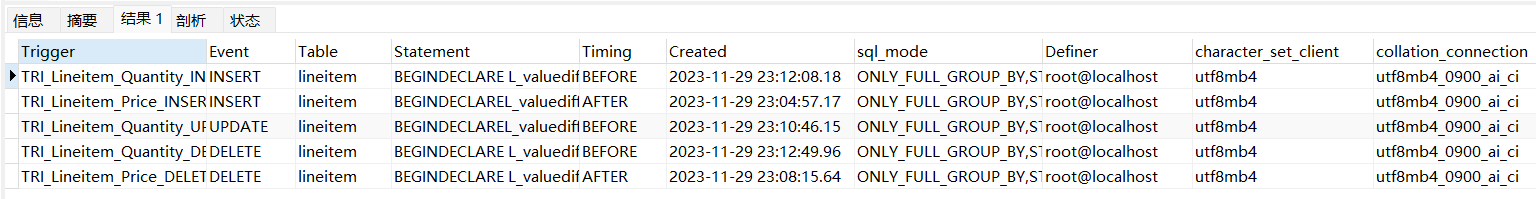
验证截图如下：



DROP TRIGGER TRI\_Lineitem\_Price\_UPDATE;

SHOW TRIGGERS;

验证截图如下：

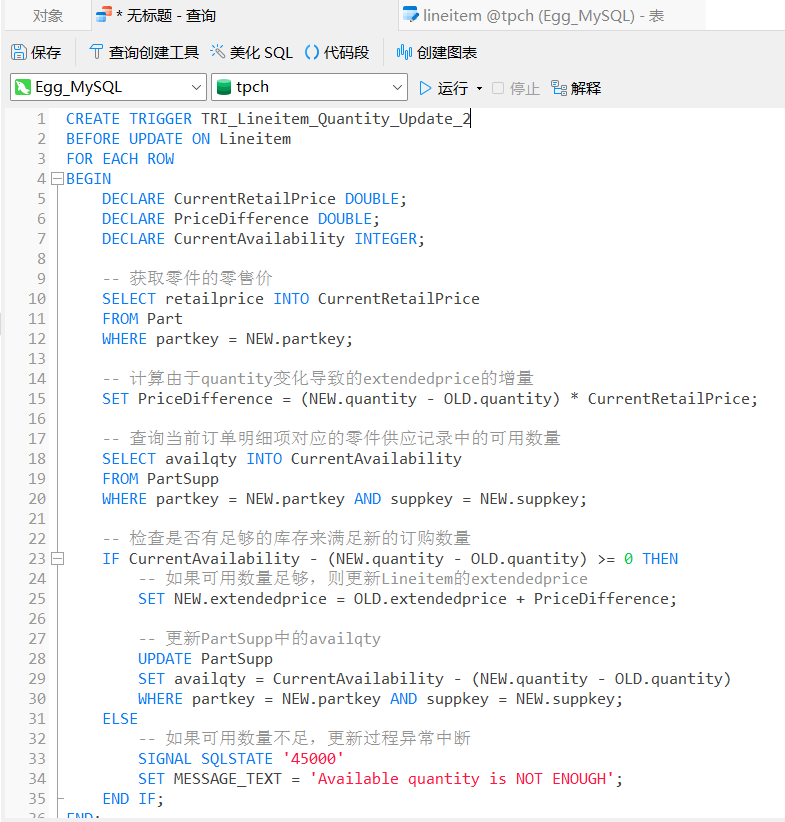


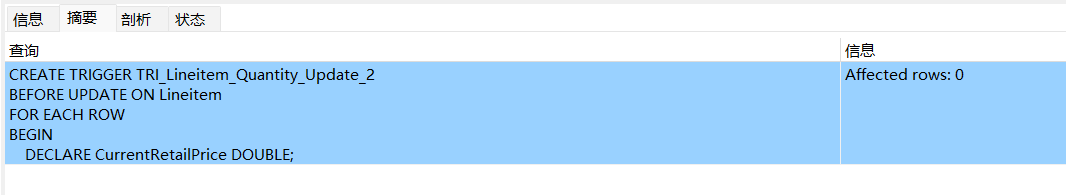
* + - 1. **与实验结果相关的文件**

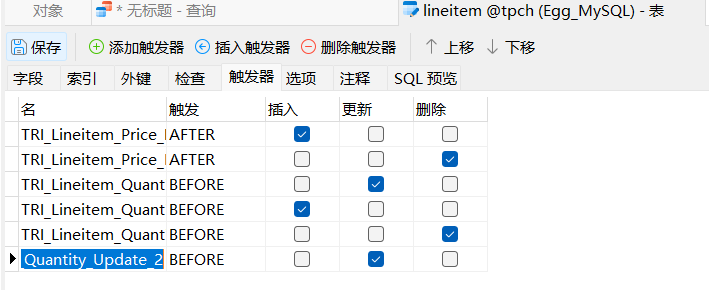
无

* + - 1. **思考题**

试设计一个BEFORE触发器， 当Lineitem表中的quantity变化时， 自动计算Lineitem表中的extendedprice值， 同时也要修改PartSupp中的availqty值(提示：extendedprice=quantity\*Part.retailprice) 。



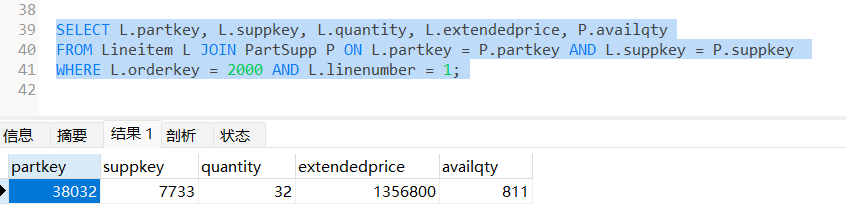




SELECT L.partkey, L.suppkey, L.quantity, L.extendedprice, P.availqty

FROM Lineitem L JOIN PartSupp P ON L.partkey = P.partkey AND L.suppkey = P.suppkey

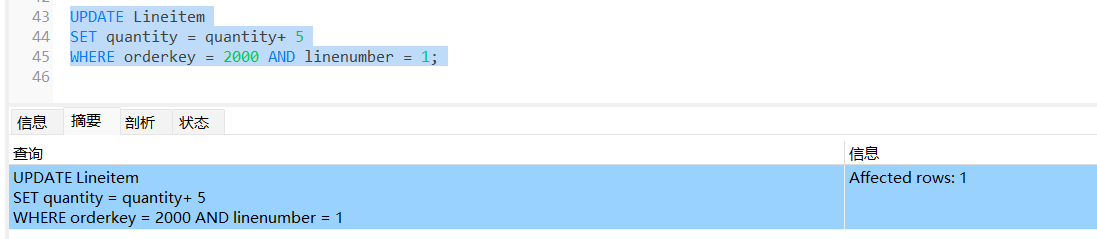
WHERE L.orderkey = 2000 AND L.linenumber = 1;



UPDATE Lineitem

SET quantity = quantity+ 5

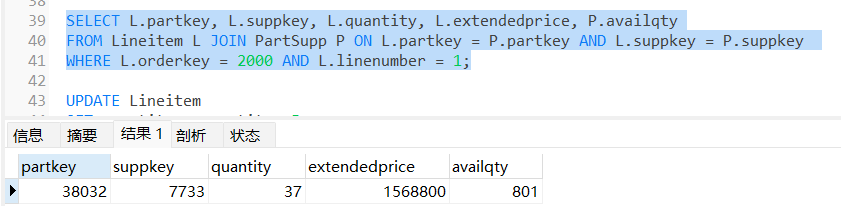
WHERE orderkey = 2000 AND linenumber = 1;



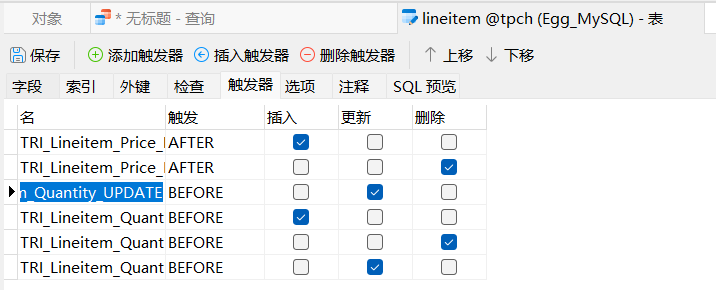
SELECT L.partkey, L.suppkey, L.quantity, L.extendedprice, P.availqty

FROM Lineitem L JOIN PartSupp P ON L.partkey = P.partkey AND L.suppkey = P.suppkey

WHERE L.orderkey = 2000 AND L.linenumber = 1;



可见成功修改extendedprice的值，quantity+=5，但availqty-=10，这是因为前一个触发器TRI\_Lineitem\_Quantity\_UPDATE没有删除，所以导致availqty多减了5



* + - 1. **实验总结**

在本次数据库实验中，我深入学习并实践了数据库触发器的设计和使用方法，深刻体会到了触发器在数据库管理和维护中的重要作用。实验中，我了解了BEFORE和AFTER类型的触发器，这两种触发器在数据更新的前后阶段发挥着关键作用，对于保证数据库的数据一致性和完整性至关重要。

通过实验，我理解了触发器在数据更新过程中自动执行特定操作的机制，这对于自动化数据管理和复杂完整性约束的实施至关重要。特别是在处理订单明细（Lineitem）和订单总价（Orders）的数据一致性问题时，触发器显示出其强大的功能。例如，当Lineitem表的价格、折扣或税率发生变化时，AFTER触发器能够自动更新Orders表的TotalPrice，以保持数据之间的一致性。

总的来说，这次实验让我加深了对数据库触发器的理解，也锻炼了我解决实际问题的能力，这次实验我明白了数据库触发器在现代数据库管理中扮演着重要的角色。