# 触发器实验

16337113

劳马东

计算机科学与技术（超算方向）

## 实验目的

掌握数据库触发器的设计和使用方法。

## 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。

## 实验环境

|  |  |
| --- | --- |
| 系统 | Windows 10 |
| SQL | MySQL 8.0 |
| 工具 | MySQL workbench |

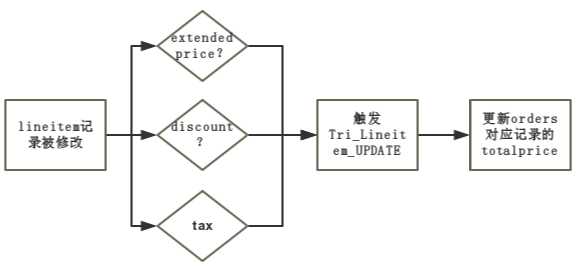
## 实验过程

###### 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order的totalprice，以保持数据一致性。

其中，totalprice=totalprice+extendedprice+(1-discount)\*(1+tax)。



DELIMITER //

CREATE TRIGGER TRI\_Lineitem\_Price\_UPDATE

AFTER UPDATE ON lineitem

FOR EACH ROW

BEGIN

DECLARE L\_valuediff REAL;

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

UPDATE orders SET totalprice = totalprice + L\_valuediff

WHERE orderkey = NEW.orderkey;

END//

MySQL语句默认以分号结束，它告诉MySQL解释器一段命令已经结束，可以开始执行。但有时候，不希望MySQL这么做，例如在函数和触发器定义中通常含有较多的语句，而这些语句以分号结束。因此，需要默认把结束符换成其他符号（如//或$$），MySQL提供了DELIMETER语句来实现这个目的。

上面的代码首先计算新的记录（NEW）的totalprice和旧记录（OLD）的totalprice的差值，然后更新orders表对应orderkey的totalprice。

1. **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。

INSERT触发器相当于extendedprice、discount、tax均改变的情况下的UPDATE触发器，所不同的是，INSERT触发器中没有OLD变量，因为插入一条新的记录之前不存在一条与其对应的旧记录。

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);

UPDATE orders SET totalprice = totalprice + L\_valuediff

WHERE orderkey = NEW.orderkey;

END//

1. **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，保持数据一致性。

DELETE触发器相当于extendedprice、discount、tax全部变成0情况下的UPDATE触发器，所不同的是，DELETE触发器中没有NEW变量，因为删除操作实际上不是更新记录，而是去掉原来的记录，因此也只能引用原来的旧记录OLD。

CREATE TRIGGER TRI\_Lineitem\_Price\_INSERT

AFTER INSERT ON lineitem

FOR EACH ROW

BEGIN

DECLARE L\_valuediff REAL;

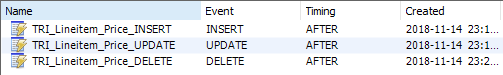
SET L\_valuediff = -OLD.extendedprice \* (1 - OLD.discount) \* (1 + OLD.tax);

UPDATE orders SET totalprice = totalprice + L\_valuediff

WHERE orderkey = OLD.orderkey;

END//

三个触发器创建成功后，在MySQL workbench中查看lineitem表的Triggers，就能发现多出了三个触发器，分别为TRI\_Lineitem\_Price\_INSERT、TRI\_Lineitem\_Price\_UPDATE和TRI\_Lineitem\_Price\_DELETE，分别对应INSERT、UPDATE和DELETE事件，都属于AFTER类型。



1. **验证触发器**

查看1号订单的含税折扣总价totalprice。

SELECT totalprice FROM orders WHERE orderkey=1;

查询结果表明orderkey为1的订单目前的总价为172799.49。



* 1. TRI\_LineItem\_Price\_UPDATE

要验证UPDATE触发器，就需要修改lineitem表中的某个记录的extendedprice、discount或tax。下面的代码修改了订单号为1的第一个明细项的税率，增加0.5%，修改其他属性同理，观察1号订单的总价变化。

UPDATE lineitem SET tax = tax + 0.005

WHERE orderkey = 1 AND linenumber = 1;

查询结果表明税率增加后，1号订单的总价提高到172918.09968，符合UPDATE触发器的定义，说明UPDATE触发器起作用了。



* 1. TRI\_LineItem\_Price\_INSERT

在lineitem表中插入一条新的记录，由于orderkey和linenumber是主键，新记录不同这两个属性不能与原有记录都相同。下面的代码在1号订单的第一个明细的基础上，将linenumber修改为100，即第100个明细。

INSERT INTO lineitem VALUES(1, 1552, 93, 100, 17, 24710.35, 0.04,

0.025, 'N', 'O', '1996-03-13', '1996-02-12', '1996-03-22', 'DELIVER IN PERSON', 'TRUCK', 'egular courts above the');

查询结果表明，1号订单的总价再次提高到197114.4744，符合INSERT触发器定义，说明INSERT触发器起作用了。



* 1. TRI\_LineItem\_Price\_DELETE

删除1号订单的第一个明细，查看它的总价变化。

DELETE FROM lineitem WHERE orderkey=1 AND linenumber=1;

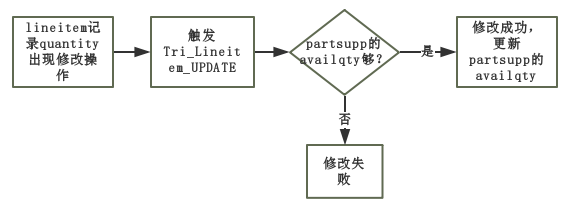
查询结果表明，1号订单的totalprice从197114.4744减少到148603.11528，符合DELETE触发器定义，说明DELETE触发器起作用。



###### BEFORE触发器

1. **在lineitem表上定义一个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。



DELIMITER //

CREATE TRIGGER TRI\_Lineitem\_Quantity\_UPDATE

BEFORE UPDATE ON lineitem

FOR EACH ROW

BEGIN

DECLARE L\_valuediff, L\_availqty INTEGER;

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

SELECT availqty INTO L\_availqty FROM partsupp

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

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

        BEGIN

UPDATE partsupp SET availqty = availqty - L\_valuediff

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

        END;

ELSE

        SIGNAL SQLSTATE '45000' SET message\_text='Available quantity is NOT ENOUGH';

END IF;

END//

在上面的代码中，首先计算新记录（NEW）的quantity和旧记录（OLD）的quantity的差值，即所需的供应数。如果partsupp中现有的供应数足够，就更新partsupp表对应partkey和suppkey的供应数，否则更新失败，向用户跑出错误信息——Available quantity is not enough。

MySQL提供了SIGNAL语句来抛出异常或错误，其中SQLSTATE后的字符串是错误代码，“00”开头代表完全成功执行，“01”开头是警告等等。“45000”是一种错误，发生这个错误时设置错误消息，就能在MySQL workbench中看到报错信息。

1. **在lineitem表上定义一个INSERT触发器**

当插入订单明细时，先检查供应表partsupp中的可用数量availqty是否足够。

定义INSERT触发器与UPDATE触发器类似，只不过没有OLD变量，或者可将OLD变量的quantity视为0。

CREATE TRIGGER TRI\_Quantity\_INSERT

BEFORE INSERT ON lineitem

FOR EACH ROW

BEGIN

DECLARE L\_valuediff, L\_availqty INTEGER;

SET L\_valuediff = NEW.quantity;

SELECT availqty INTO L\_availqty FROM partsupp

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

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

BEGIN

UPDATE partsupp SET availqty = availqty - L\_valuediff

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

END;

ELSE

SIGNAL SQLSTATE '45000' SET message\_text='Available quantity is NOT ENOUGH';

END IF;

END//

1. **在lineitem表上定义一个DELETE触发器**

当删除订单明细时，该订单明细订购的数量要归还对应的零件供应记录。

DELETE触发器的定义省去了判断是否足够的过程，因为回收订单时availqty必然是增加的。

CREATE TRIGGER TRI\_Lineitem\_Quantity\_DELETE

BEFORE DELETE ON lineitem

FOR EACH ROW

BEGIN

    DECLARE L\_valuediff, L\_availqty INTEGER;

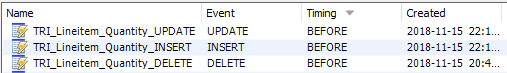
SET L\_valuediff = -OLD.quantity;

UPDATE partsupp SET availqty = availqty - L\_valuediff

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

END//

在MySQL workbench中查看lineitem的Triggers，发现多出了3个BEFORE触发器，分别是TRI\_Lineitem\_Quantity\_UPDATE、TRI\_Lineitem\_Quantity\_INSERT和TRI\_Lineitem\_Quantity\_DELETE，分别对应UPDATE、INSERT和DELETE操作。



1. **验证触发器**

查看1号订单第一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 = 1 AND L.linenumber=1;

下图表明1号订单还有7030个可用，其中第一个明细项订购了17个，。



* + 1. TRI\_LineItem\_Quantity\_UPDATE

修改1号订单第一个明细项的订购数量，增加5个。

UPDATE lineitem SET quantity = quantity + 5

WHERE orderkey = 1 AND linenumber = 1;

再次查询，结果显示1号订单只剩7025个可用，第一个明细项的订购数量也变成了22个。



假如我们增加10000个订购，会发现更新失败，MySQL workbench显示报错，信息为“Available quantity is NOT ENOUGH”，符合UPDATE触发器定义。

UPDATE lineitem SET quantity = quantity + 10000

WHERE orderkey = 1 AND linenumber = 1;



再次查询，发现订购数量依然是22，可用数量也没变。

* + 1. TRI\_LineItem\_Quantity\_INSERT

在lineitem中插入一条orderkey为1、linenumber为100、订购数为17的订单。

INSERT INTO lineitem VALUES(1, 1552, 93, 100, 17, 24710.35, 0.04, 0.025,

'N', 'O', '1996-03-13', '1996-02-12', '1996-03-22', 'DELIVER IN PERSON', 'TRUCK', 'egular courts above the');



同理，插入一条订购数为10000的记录，报错显示可用数量不够，查看lineitem表没有这条记录，partsupp的1号订单可用数量依然为7008.

INSERT INTO lineitem VALUES(1, 1552, 93, 101, 10000, 24710.35, 0.04, 0.025,

'N', 'O', '1996-03-13', '1996-02-12', '1996-03-22', 'DELIVER IN PERSON',

'TRUCK', 'egular courts above the');



* + 1. TRI\_LineItem\_Quantity\_DELETE

删除lineitem中1号订单的第100个明细，查询partsupp中的可用数量，又变回了7025，同时第100个明细不再存在于lineitem中。

DELETE FROM lineitem WHERE orderkey = 1 AND linenumber = 100;

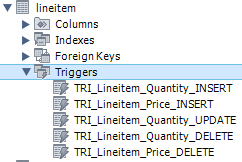


###### 删除触发器

删除触发器TRI\_Lineitem\_Price\_UPDATE。

DROP TRIGGER TRI\_Lineitem\_Price\_UPDATE;

查看lineitem表所有的Trigger，发现没有了TRI\_Lineitem\_Price\_UPDATE。



## 实验总结

触发器实验让我想起了基于事件的编程方法（例如图形界面设计中的按钮事件），当一件事情发生时，就回调一个函数。触发器正是这样的一种回调机制，当表中发生UPDATE、INSERT、DELETE操作时，对应的函数被执行。函数中的逻辑包括信息打印、数据同步等。在AFTER和BEFORE触发器中出现了数据同步的操作；在BEFORE触发器抛出了异常，以向用户反馈错误信息。

这个实验同样也遇到了一些问题。

首先，MySQL的触发器并不针对表的某些属性。即定义触发器时不需用OF语句，MySQL会自动将触发器定义中涉及到的属性作为触发器关联的属性。为了证明这个猜想，我专门在AFTER UPDATE触发器上做了测试，修改除extendedprice、discount、tax外的其他属性，观察是否输出信息。结果没有输出任何信息，说明触发器没有被触发。

其次，触发器定义前需要修改语句的结束符。一开始，我按照书上代码输入时，MySQL workbench编辑器给我的代码加上了红色波浪线，我就知道问题来了。原来，默认的语句块结束符是分号，而函数体中有许多以分号结尾的语句。于是我查看MySQL 8.0官方文档，发现要更改DELIMETER为//，这样整个BEGIN...END语句块的结束符就变成了//，也就可以在语句块中输入以分号结尾的语句，而不会让MySQL解释器认为语句块以BEGIN开始而以一个不匹配的语句结束。

最后，书上的代码有一些逻辑上的错误。例如，在定义INSERT触发器时，是不存在OLD变量的，因为这是一个全新的记录，没有与之对应的原记录；定义DELETE触发器时，不存在NEW变量，因为删除一条记录不涉及产生新记录。