实验名称	触发器实验
实验类型	■ 验证 □ 综合 □ 设计 □ 创新
实验日期	4 月 29 日
实验成绩	

一、 实验目的

掌握数据库触发器的设计和使用方法.

二、实验要求

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.

三、实验内容

(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。

```
use tpch;
CREATE TRIGGER TRI_Lineitem_Price_UPDATE
    after update
    ON lineitem
    FOR EACH ROW
BEGIN
    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;
```

Trigger	Event	Table	Statement
TRI_Lineitem_Price_UPDATE	UPDATE	lineitem	BEGIN

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
    set @L_valuediff = NEW.extendedprice * (1 - NEW.discount) * (1 + NEW.tax);
    Update orders set totalprice = totalprice + @L_valuediff where orderkey = NEW.orderkey;
END;
```

```
TRI_Lineitem_Price_UPDATE  UPDATE  lineitem  BEGIN  AFTER
```

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
    set @L_valuediff = - old.extendedprice * (1 - old.discount) * (1 + old.tax);
    Update orders set totalprice = totalprice + @L_valuediff where orderkey = old.orderkey;
END;
```

```
TRI_Lineitem_Price_DELETE  DELETE  lineitem  BEGIN  AFTER
```

4. 验证触发器 TRI Lineitem_ Price_UPDATE。

```
select totalprice
from orders
where orderkey = 5;
```

totalprice
13.42

```
update lineitem
set tax = tax + 0.05
where orderkey = 5
  and linenumber = 5;
```

```
update lineitem                    Affected rows: 0, Time: 0.004000s
```

```
select totalprice
from orders
where orderkey = 5;
```

totalprice
13.42

(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
    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;
    end if;
END;
```

```
CREATE TRIGGER TRI_Lineitem_Quantity_UP...    Affected rows: 0, Time: 0.007000s
```

2. 在 Lineitem 表上定义一个 BEFORE INSERT 触发器,当插人订单明细,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
    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
            #MySQL没有rasie notice提示，用select代替（在触发器中不能执行select，于是删除）
            UPDATE partsupp
            set availqty = availqty - @L_valuediff
            where partkey = NEW.partkey
              and suppkey = NEW.suppkey;
        end;
    end if;
END;
```

```
CREATE TRIGGER TRI_Lineitem_Quantity_INS...    Affected rows: 0, Time: 0.006000s
```

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
    set @L_valuediff := -OLD.quantity;
    update partsupp
    set availqty = availqty - @L_valuediff
    where partkey = OLD.partkey
      and suppkey = OLD.suppkey;
END;
```

```
CREATE TRIGGER TRI_Lineitem_Quantity_DE...    Affected rows: 0, Time: 0.008000s
```

4. 验证触发器 TRI_Lineitem_Quantity_UPDATE.

```
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;
```

partkey	suppkey	quantity	availqty
1	99	67.99	21

```
update lineitem
set quantity = quantity + 5
where orderkey = 1
  and linenumber = 1;

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;
```

partkey	suppkey	quantity	availqty
1	99	72.99	16

(3) 删除触发器

删除触发器 TRI_Lineitem_Price_UPDATE.

```
drop trigger TRI_Lineitem_Price_INSERT;
drop trigger TRI_Lineitem_Quantity_INSERT;
drop trigger TRI_Lineitem_Price_UPDATE;
drop trigger TRI_Lineitem_Quantity_UPDATE;
drop trigger TRI_Lineitem_Price_DELETE;
drop trigger TRI_Lineitem_Quantity_DELETE;
```

四、实验总结

通过本次实验，学习使用了触发器的设计和使用方法，对相关知识点掌握更加牢固。

本次实验中，我原本使用的是终端运行 mysql 进行实验，但是在前几个指令中，他会将一条指令视为多条指令运行，在进行一系列查询之后决定使用客户端进行实验。

更改之后，对每一个语句都能完美运行。

五、思考讨论

Create trigger TRI_Lineitem_Quantity_UPDATE

After update on lineitem

for each row

```
        as
begin
        declare L_valuediff integer;
        set L_valuediff = new.quantity – old.quantity;
        update lineitem, part
        set lineitem.extendedprice = lineitem.extendedprice * part.retailprice
        where lineitem.partkey = part.partkey;
        update partsupp
        set availqty = availqty – L_valuediff
        where partkey = new.partkey and suppkey = new.suppeky;
end;
```

六、参考资料

《数据库系统概论》第 5 版，高等教育出版社