* + 1. 实验二：触发器的创建与测试

**功能：**

检查订单明细表Sales.SalesOrderDetail中的信息，如果修改记录中的产品单价UnitPrice大于产品公开报价（Production.Product.ListPrice），则不能进行修改并抛出错误信息，否则，进行修改并将修改的有关信息写到Production.ProuctUpdateLog表中。

**要求：**

1. 使用RAISEERROR抛出错误信息。
2. 修改信息记录表Production.ProductUpdateLog的内容：记录编号、订单编号、订单明细编号、产品编号、产品的公开报价、修改前产品的单价、修改后产品的单价、修改者的登录名。使用存储过程完成该功能，并在存储过程中调用该存储过程。
3. 给出触发器和存储过程的源代码和简要的说明（可以在代码中使用注释进行说明）。
4. 设计触发器测试方案并给出测试的命令和结果，必要时可对测试结果进行分析。

2 源代码及其说明

-- 指定数据库

use AdventureWorks;

go

-- 创建指定用表 Production.Productupdatelog

create table Production.Productupdatelog(

id int identity(1,1) primary key, -- 记录编号

SalesOrderID int, -- 订单编号

SalesOrderDetailID int, -- 订单明细编号

ProductID int, -- 产品编号

ListPrice money, -- 产品的公开报价

UnitPrice\_old money, -- 修改前产品的单价

UnitPrice\_new money, -- 修改后产品的单价

name char(20) --修改者的登录名

)

-- create the table of the Production.Productupdatelog

-- 创建存储过程

-- create the procedure of the insert data into the Production.Productupdatelog

/\*

1. 使用SYSTEM\_USER系统函数获取我们的当前的登陆的名称

2. 记录编号使用IDENTITY(1,1)自动生成

3. 订单编号，订单明细编号，产品编号,修改前的产品的单价是我们的 Salse.SalesOrderDetail表中的列数据

4. 修改后的产品的单价从我们的触发器inserted表中获取

5. 产品的公开报价来自于我们的表Production.ProductListPriceHistory的 ListPrice列

\*/

create proc sp\_productupdatelog

@SalesOrderID int,

@SalesOrderDetailID int,

@ProductID int,

@UnitPrice\_old money,

@UnitPrice\_new money,

@ListPrice money

as

insert into Production.Productupdatelog values(

@SalesOrderID,

@SalesOrderDetailID,

@ProductID,

@ListPrice,

@UnitPrice\_old,

@UnitPrice\_new,

SYSTEM\_USER

)

GO

-- create the procedure of the insert data into the Production.Productupdatelog

-- create trigger for the table Sales.SalesOrderDetail of update or insert

-- create trigger for the table Sales.SalesOrderDetail of update or insert

-- 创建UPDATE触发器

CREATE TRIGGER defend\_update ON Sales.SalesOrderDetail

INSTEAD OF UPDATE

AS

BEGIN

DECLARE @unitprice\_old money;

DECLARE @judge\_unitprice money; -- 声明单价变量

DECLARE @judge\_listprice money; -- 声明公开的价格变量

SELECT @judge\_unitprice = UnitPrice FROM inserted;

SELECT @judge\_listprice = ListPrice FROM Production.ProductListPriceHistory AS P, inserted WHERE P.ProductID = inserted.ProductID;

SELECT @unitprice\_old = S.UnitPrice FROM Sales.SalesOrderDetail AS S, inserted WHERE inserted.SalesOrderDetailID = S.SalesOrderDetailID AND inserted.SalesOrderID = S.SalesOrderID;

IF @judge\_unitprice > @judge\_listprice

BEGIN

RAISERROR(

'TWO HIGH , CAN NOT INSERT THE DATE',

16,

1

)

ROLLBACK TRANSACTION

END

ELSE

-- update the data into the Sales.SalesOrderDetail table

-- update the data into the Production.Productupdatelog table

DECLARE @SalesOrderID int;

DECLARE @SalesOrderDetailID int

DECLARE @ProductID int

DECLARE @UnitPrice\_new money

DECLARE @ListPrice money

SELECT @SalesOrderID = inserted.SalesOrderID,@SalesOrderDetailID = inserted.SalesOrderDetailID ,@ProductID = inserted.ProductID,@UnitPrice\_new = inserted.UnitPrice,@ListPrice = P.ListPrice FROM inserted,Production.ProductListPriceHistory AS P

UPDATE Sales.SalesOrderDetail SET Sales.SalesOrderDetail.UnitPrice = @judge\_unitprice

WHERE Sales.SalesOrderDetail.ProductID = @ProductID;

-- exec the procedure which insert the data into the productupdatelog

EXEC sp\_productupdatelog @SalesOrderID , @SalesOrderDetailID , @ProductID ,@unitprice\_old ,@UnitPrice\_new , @ListPrice

END

GO

-- 测试用例，对我们的用例进行修改

-- test the trigger

-- select \* from Sales.SalesOrderDetail where ProductID = 707;

-- select \* from Production.ProductListPriceHistory where SalesOrder;

-- select \* from Production.Productupdatelog;

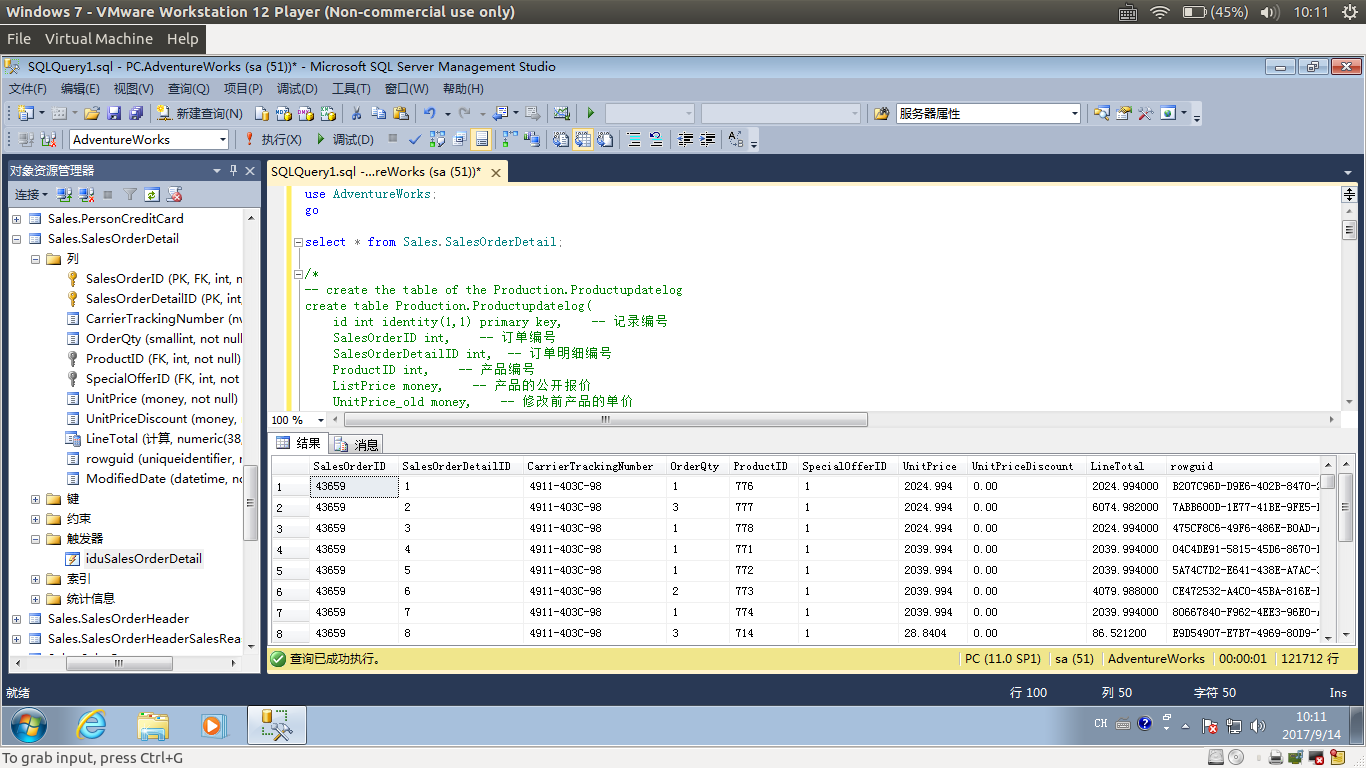
-- update Sales.SalesOrderDetail set UnitPrice = 10 where ProductID = 707;

1. 测试方案以及测试结果
2. 登录账户

使用sa登录账户并打开一个新的查询，选择进入AdventureWorks

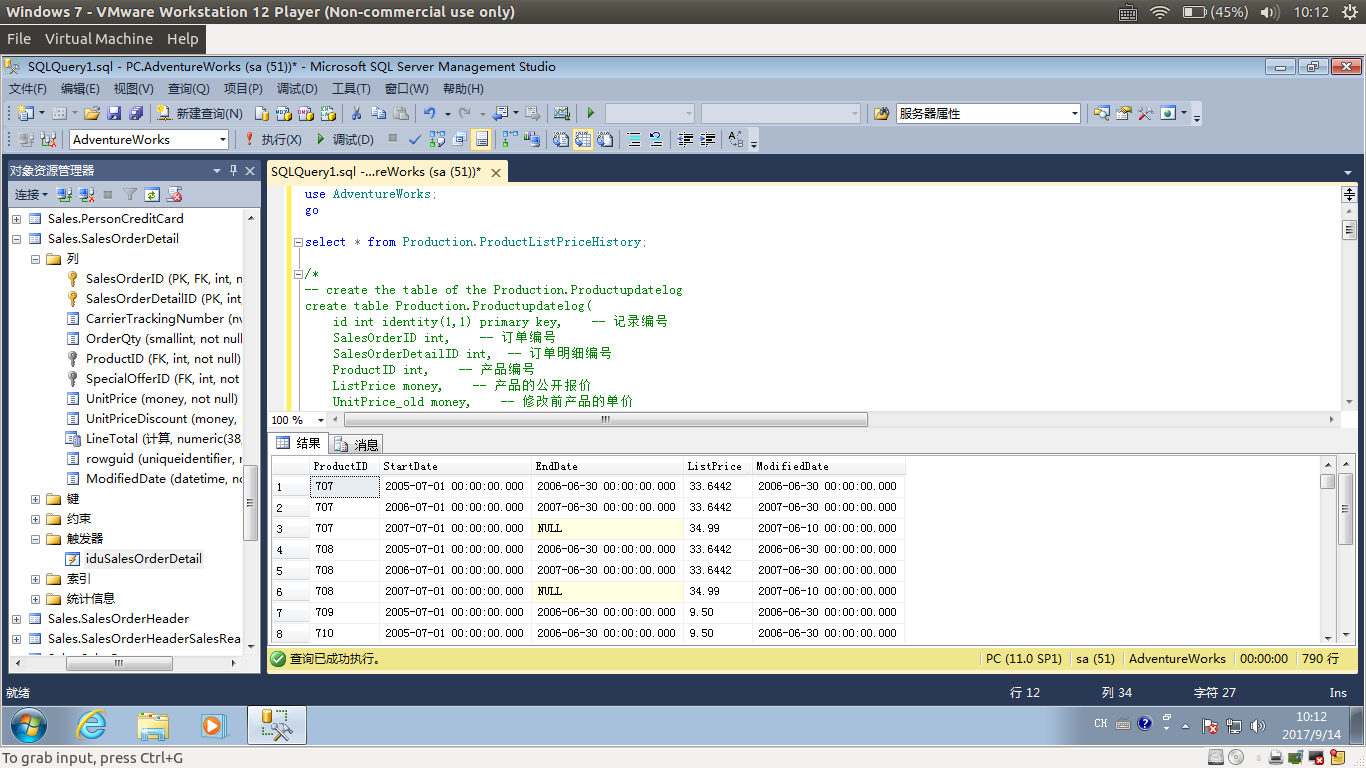
数据库，查询我们已有的Sales.SalesOrderDetail表中的数据，以便

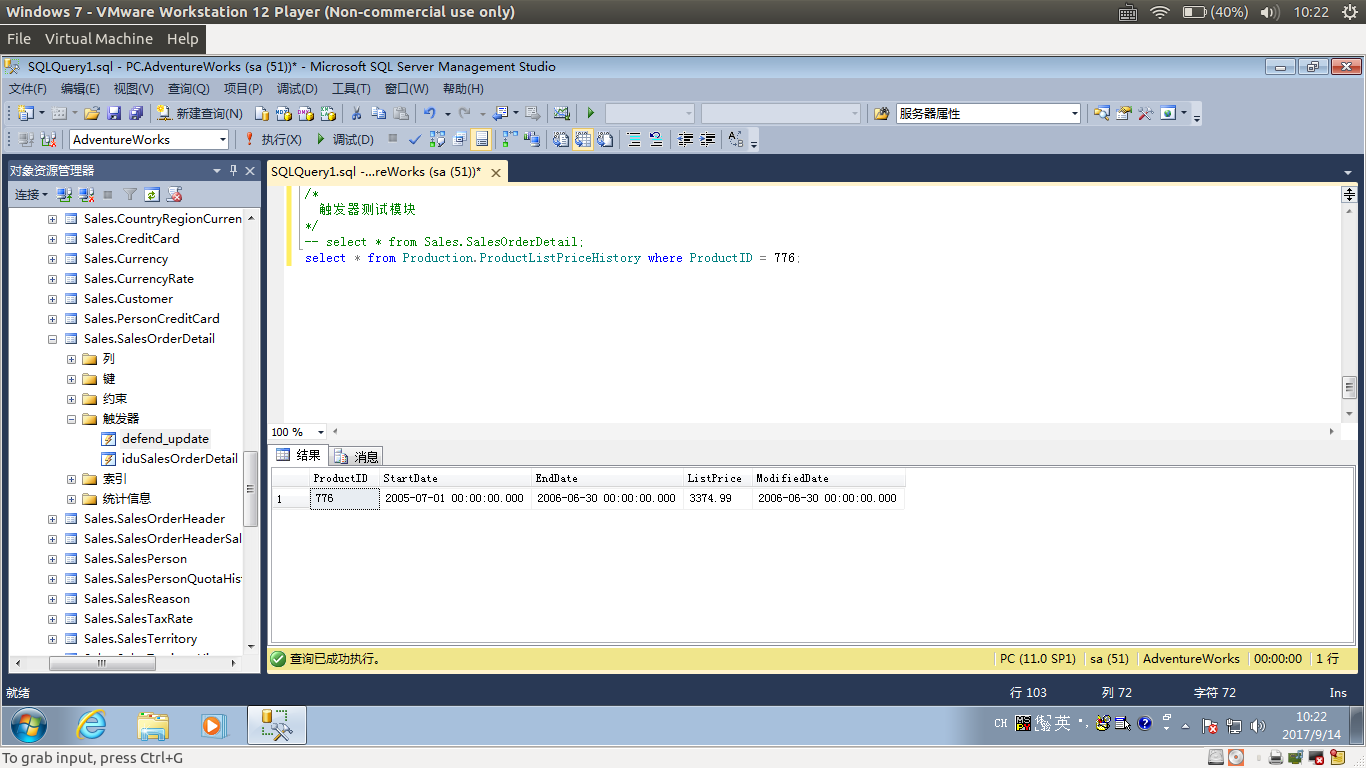
我们之后根据显示的内容确定之后要进行测试的目标



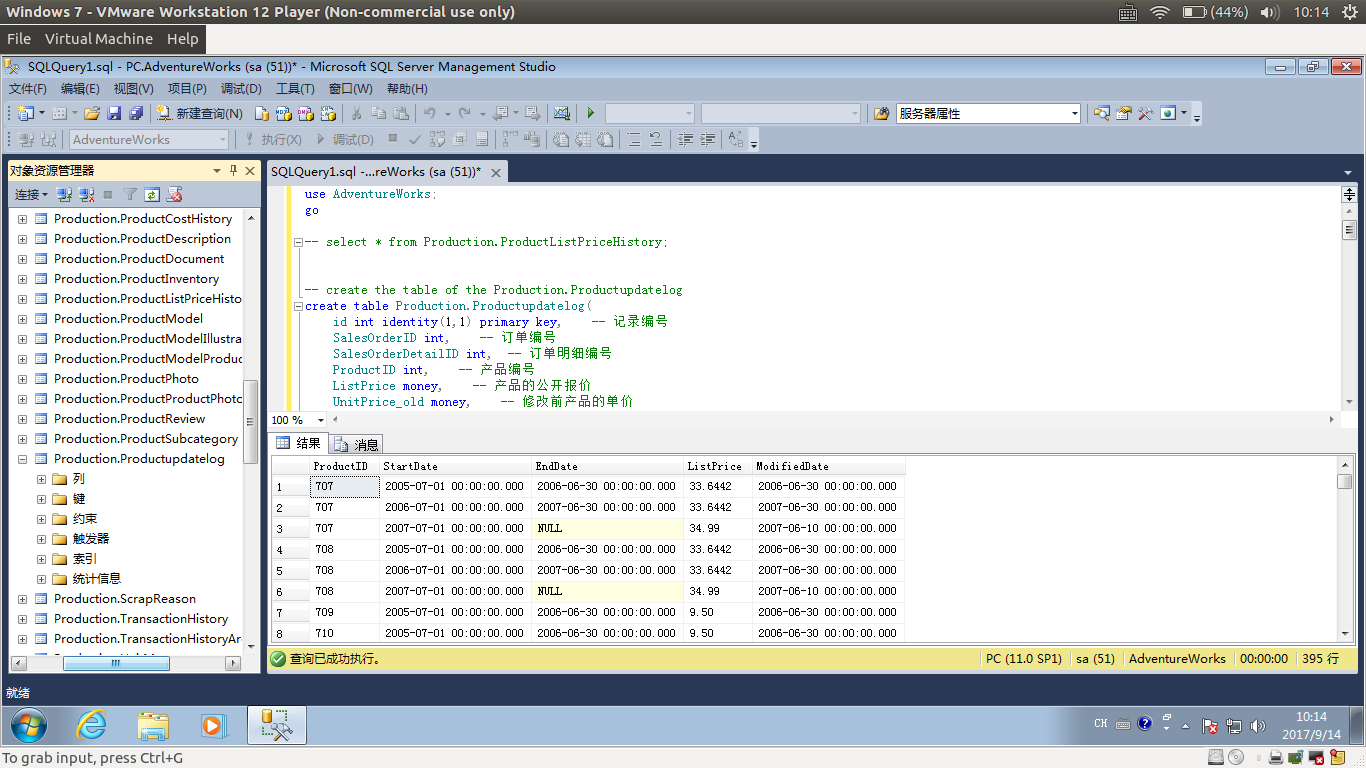
目前首先决定我们要测试的产品的详细订单编号是1

2. 查询Product.Production表的数据，进一步选定目标



进一步决定我们的详细的测试目标是编号为776号的产品，根据上图的查询结果我们确定了编号776的产品的公开报价是3374.99，如下图所示

3. 创建log表

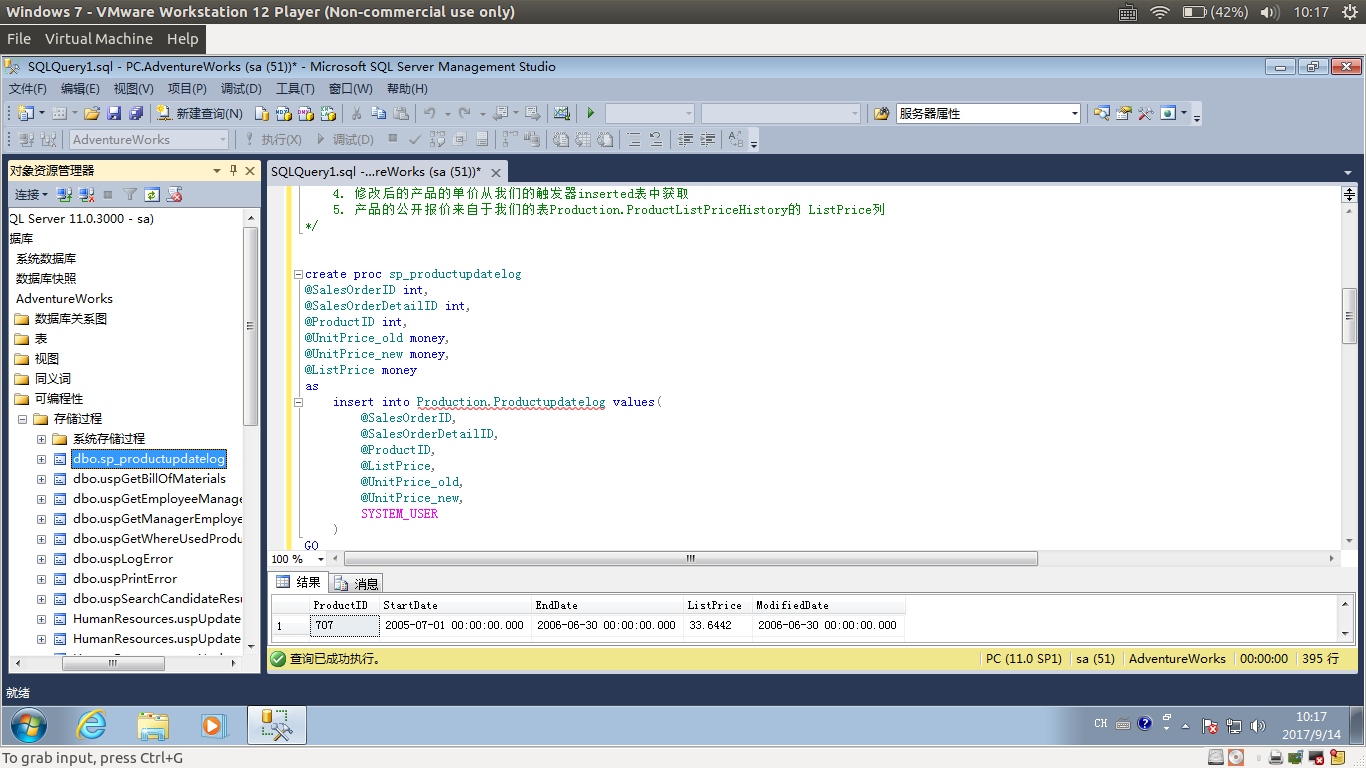


按照要求创建log表

记录表Production.ProductUpdateLog的内容：记录编号、订单编号、订单明细编号、产品编号、产品的公开报价、修改前产品的单价、修改后产品的单价、修改者的登录名

表中的数据采用INDENTITY作为自增长主键

4.创建针对表Sales.SalesOrderDetail的触发器的前置步骤，创建针对我们新建的log表的插入的存储过程，存储过程实现对我们的log表的插入操作

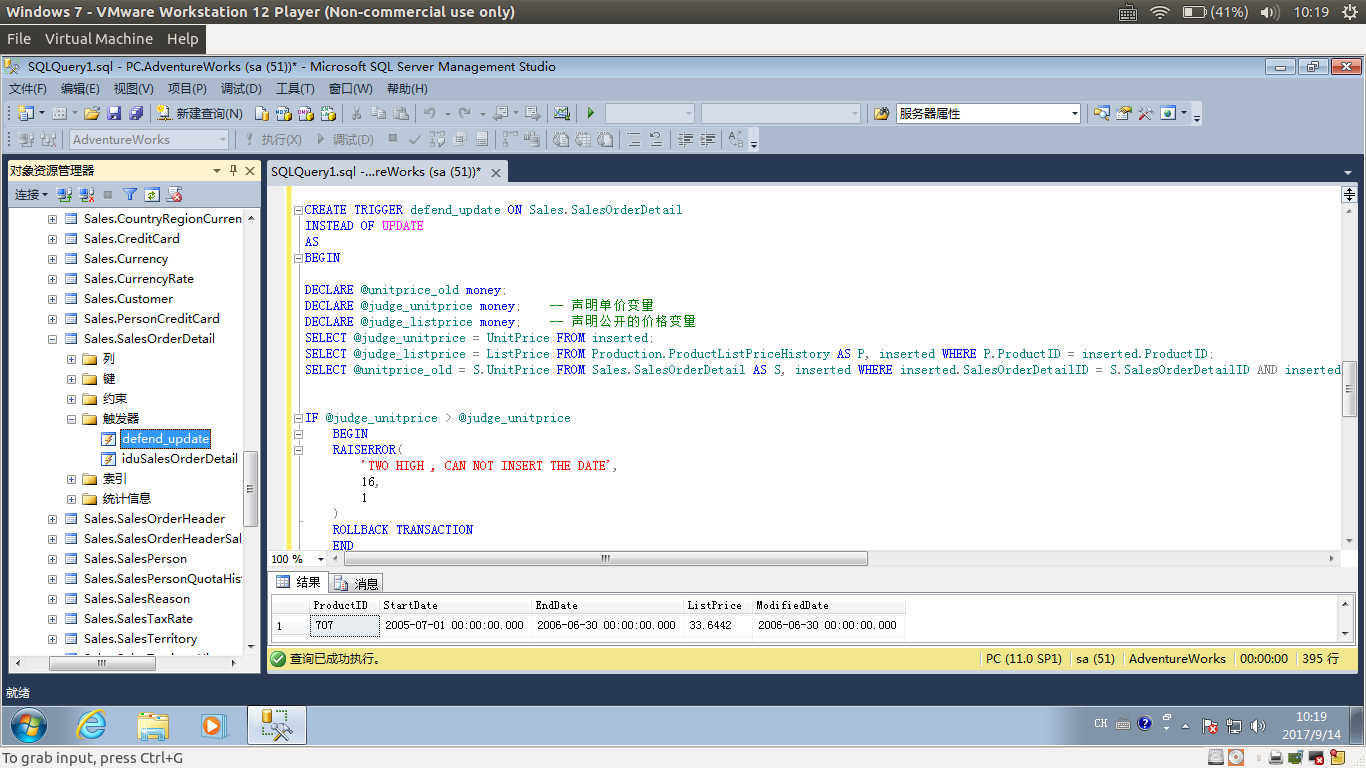


5.创建触发器

创建针对Sales.SalesOrderDetail表的触发器，该处罚期是INSTEAD OF触发器，是一个针对UPDATE更新操作的触发器

功能实现了:

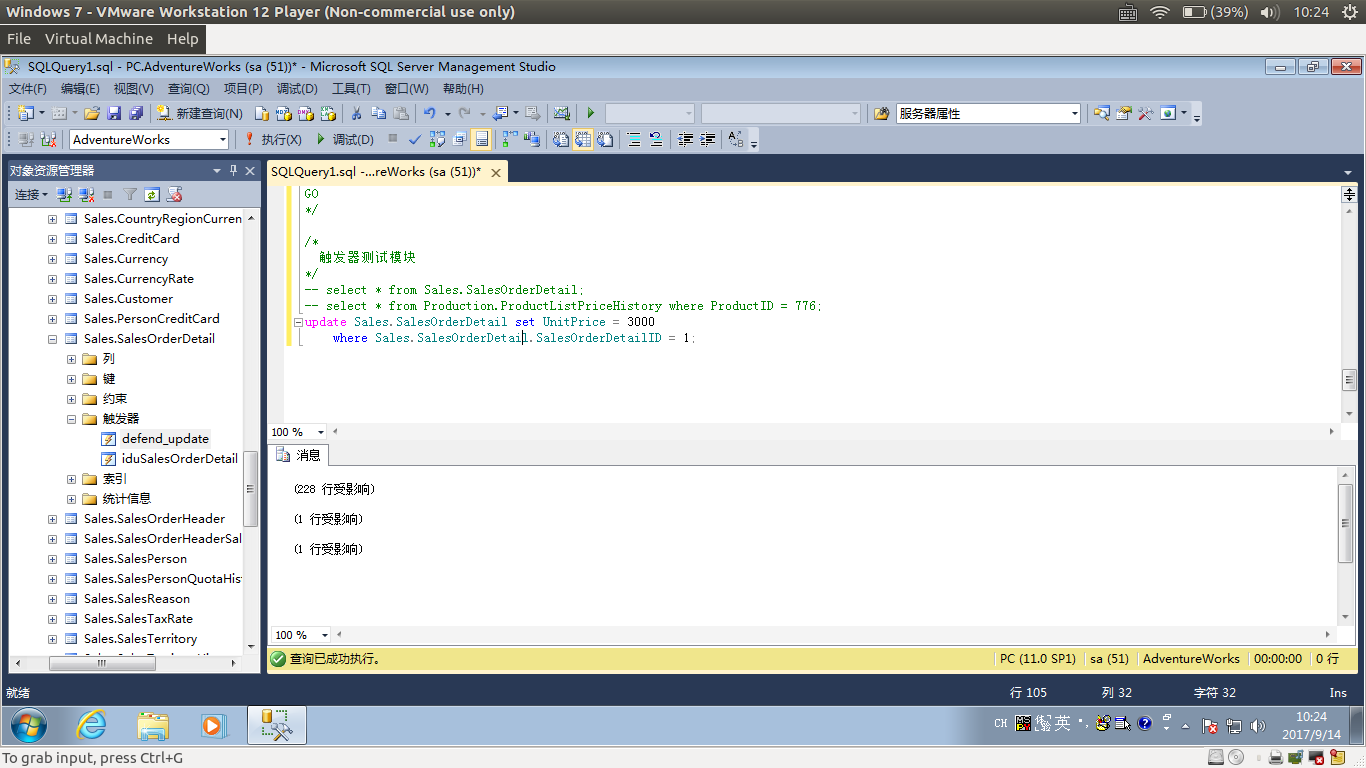
检查订单明细表Sales.SalesOrderDetail中的信息，如果修改记录中的产品单价UnitPrice大于产品公开报价（Production.Product.ListPrice），则不能进行修改并抛出错误信息，否则，进行修改并将修改的有关信息写到Production.ProuctUpdateLog表中。

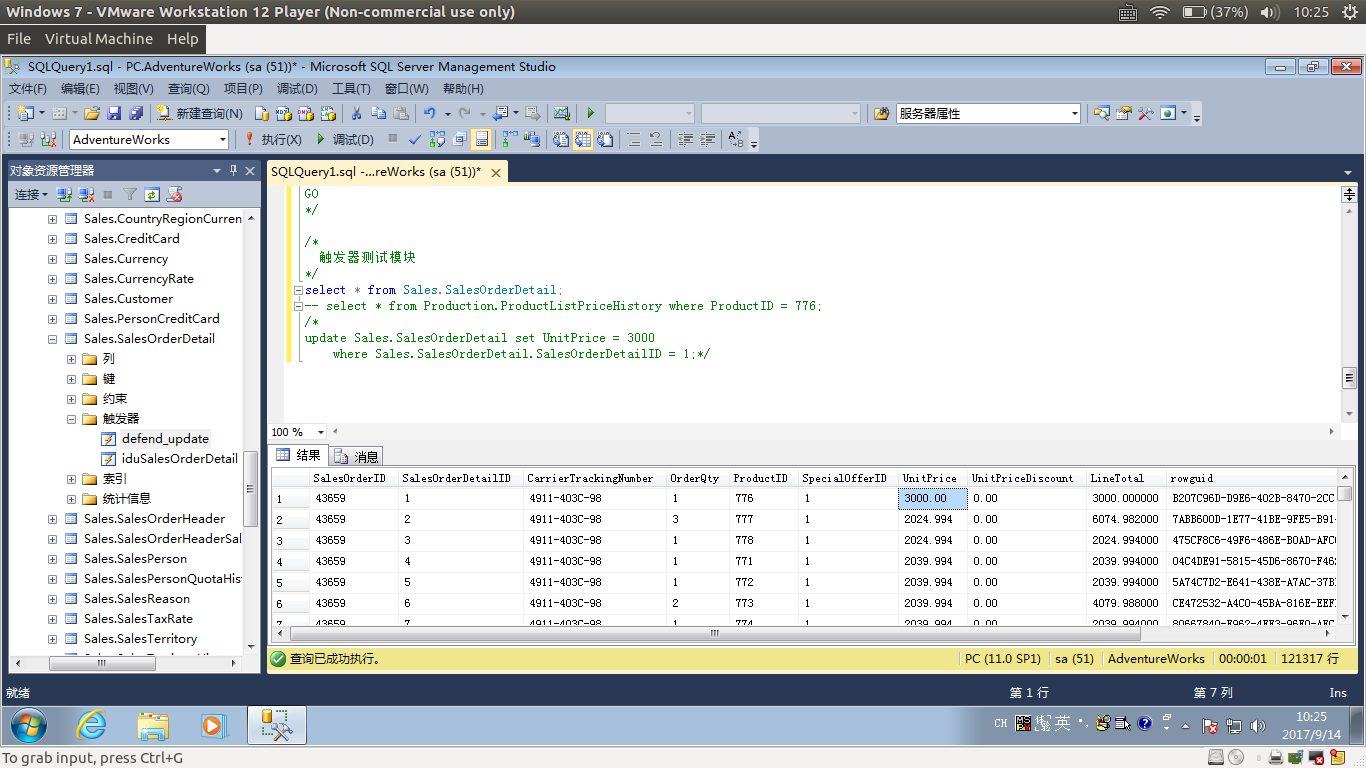


6.开始测试

6.1 测试触发器的正常执行的语句

首先我们将器单价更新成3000,因为3000小于公开报价3374.99所以实惠执行我们的正确的更新操作

结果现实确实将数据更新到3000，触发器功能正确



6.2异常执行

我们将我们的单价更改为8000远远大于我们的公开报价的3374.99，按照触发器的定义语句来看，应该会抛出一个16级状态是1的异常，异常大打印语句是

消息 50000，级别 16，状态 1，过程 defend\_update，第 71 行

TWO HIGH , CAN NOT INSERT THE DATE

检测结果也是符合的，表明触发器的功能正确

