

**数据库报告**

|  |  |
| --- | --- |
| 题目 | **实验：数据控制** |
| 姓名 |  |
| 学号 |  |
| 班级 |  |
| 指导教师 | 肖达 |

目录

[实验：数据控制 2](#_Toc134308398)

[实验目的 2](#_Toc134308399)

[实验工具 2](#_Toc134308400)

[实验数据库 2](#_Toc134308401)

[实验内容 3](#_Toc134308402)

[1.安全性部分: 3](#_Toc134308403)

[2.完整性部分: 17](#_Toc134308404)

[3.备份与恢复: 23](#_Toc134308405)

[4.触发器 27](#_Toc134308406)

[总结 30](#_Toc134308407)

# 实验：数据控制

## 实验目的

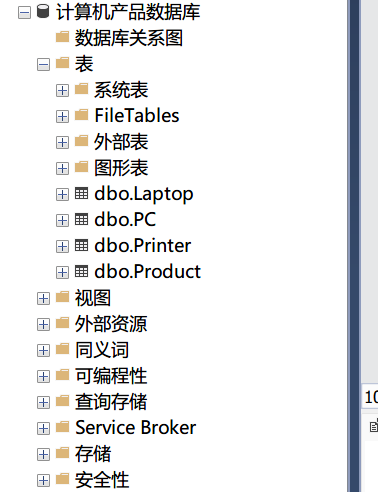
熟悉通过SQL对数据库进行数据控制，包括安全性、完整性和数据库恢复

## 实验工具

SQL Server/MySQL提供的交互查询工具

## 实验数据库

交互式SQL所用数据库或自行设计（至少三个表）

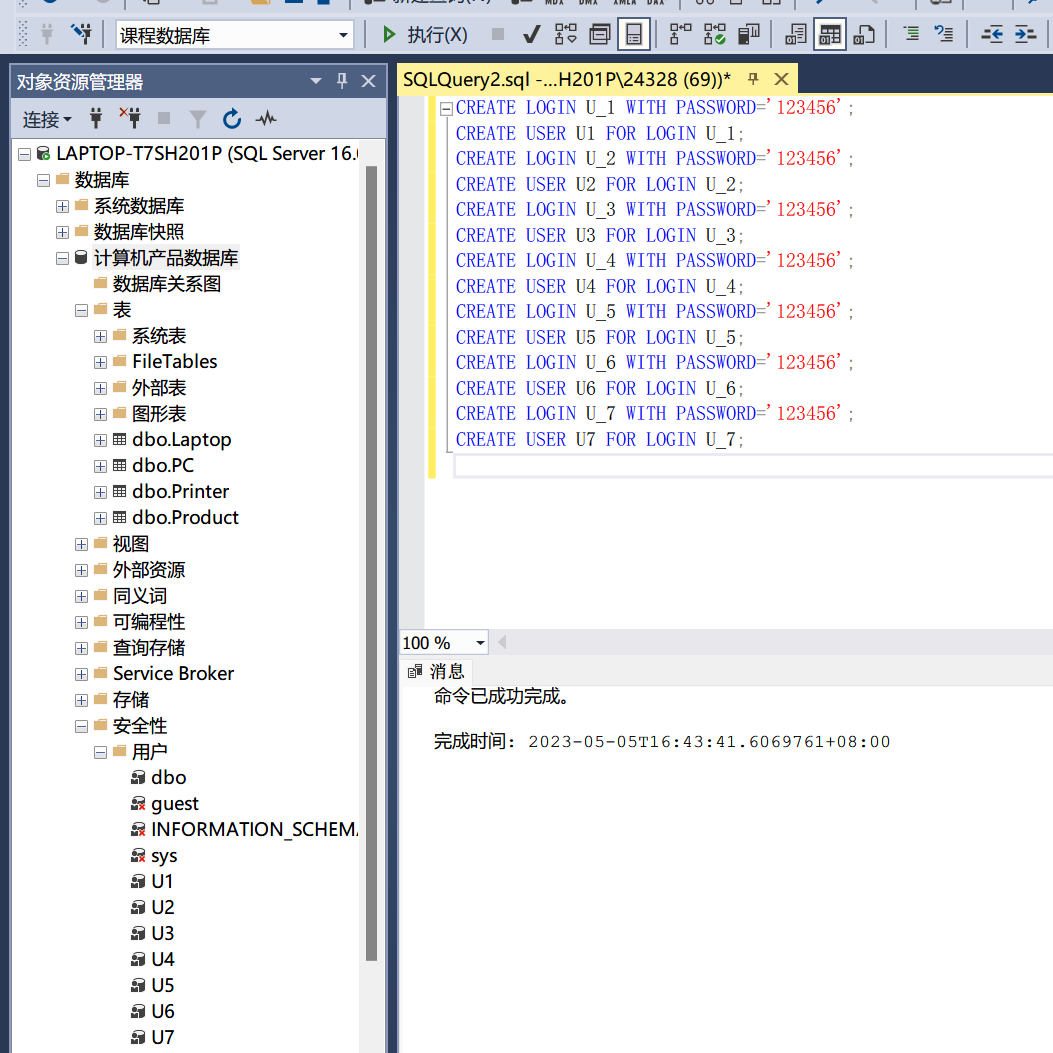


## 实验内容

### 1.安全性部分:

授权与回收

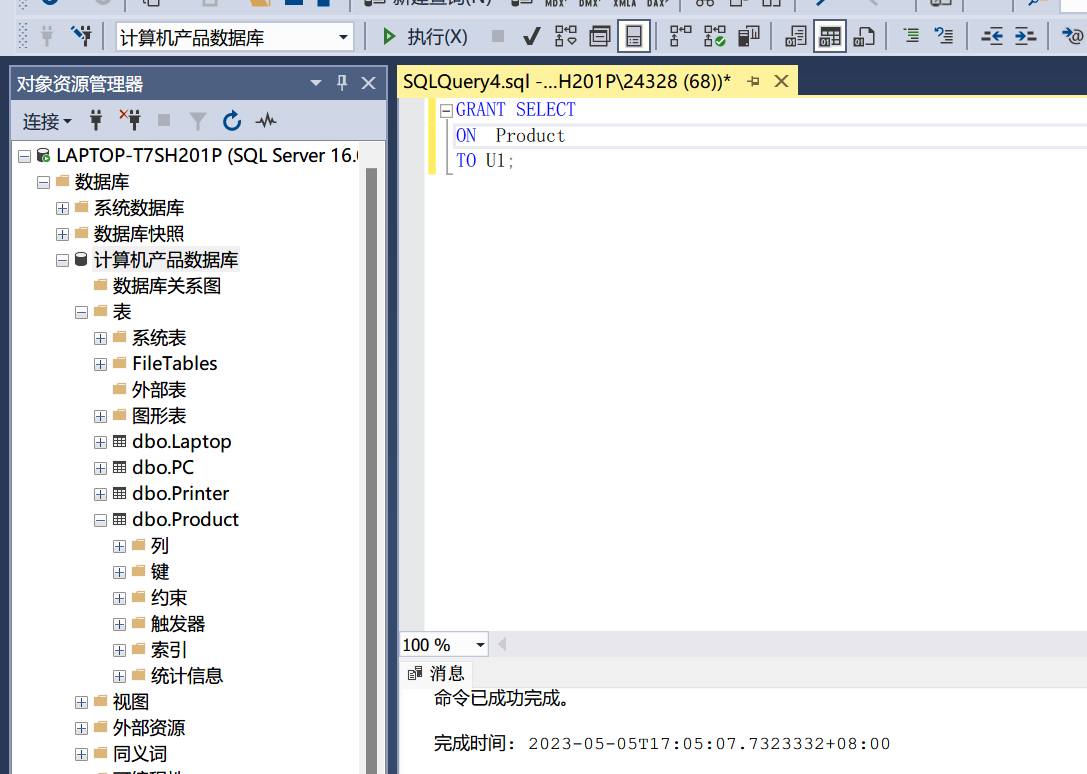
1. 在数据库中由DBA创建若干用户，权限全部选择为CONNECT（SQL Server中的db\_accessadmin角色）

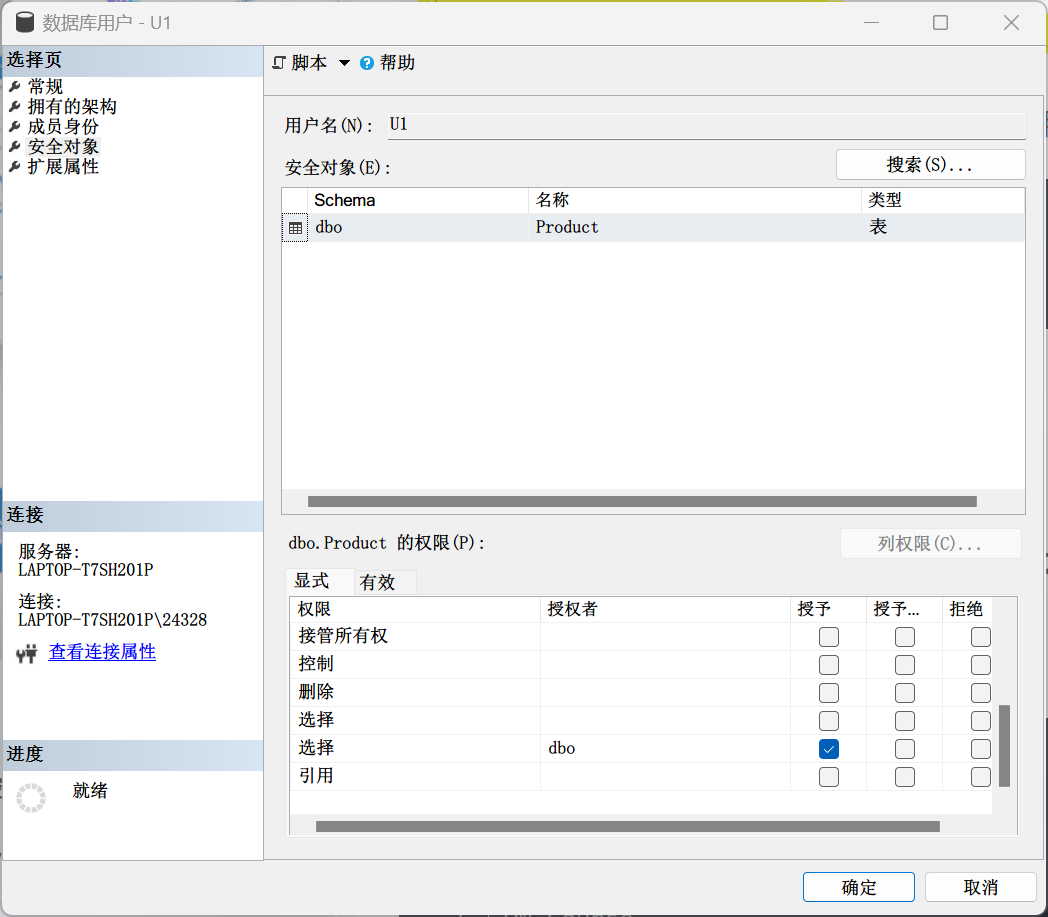


(2)仿照教材4.2.4 [例1]~[例10]，在DBA与这些用户之间进行授权和回收，并查看效果（4用例）

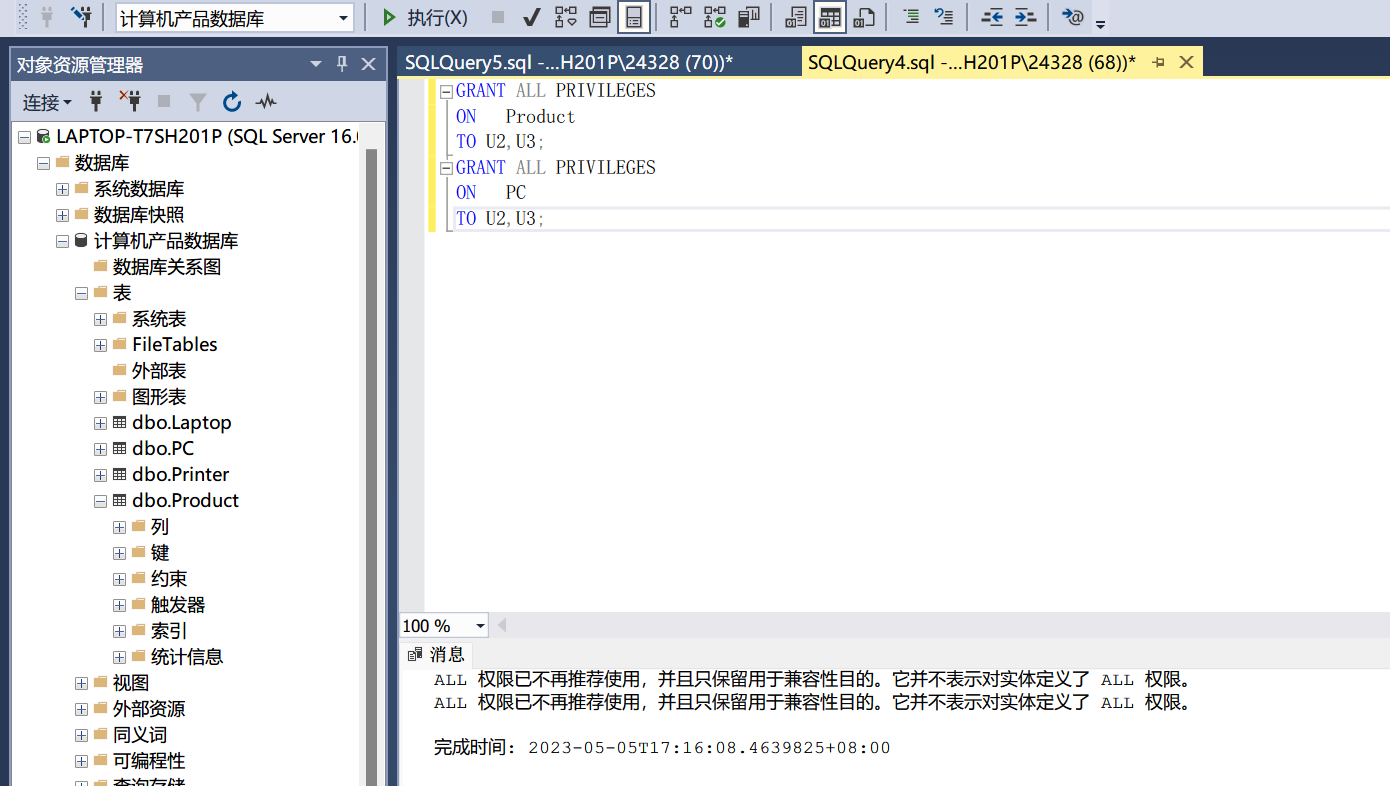
注意SQL Server中登录名与用户的区别

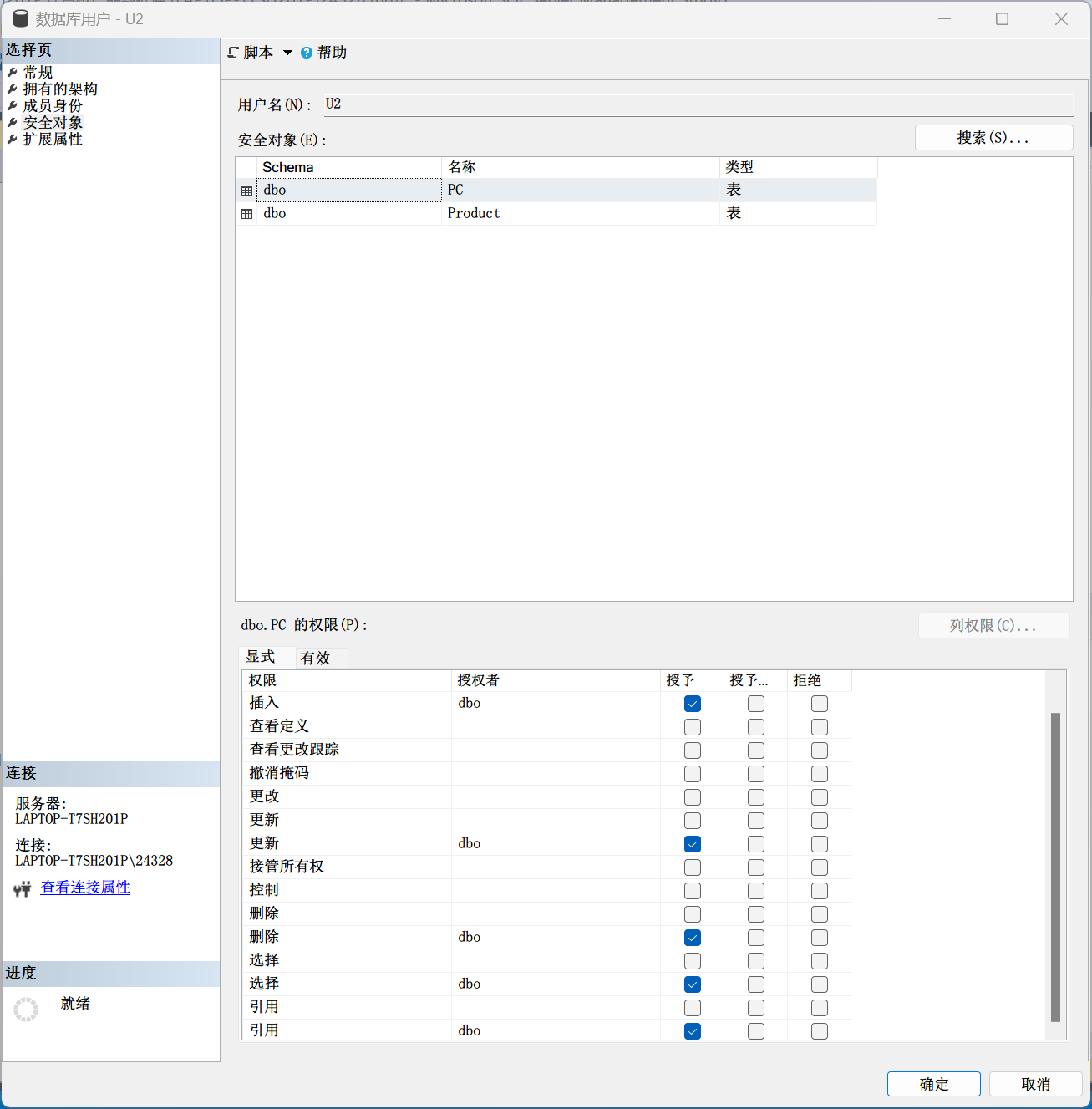
例1：



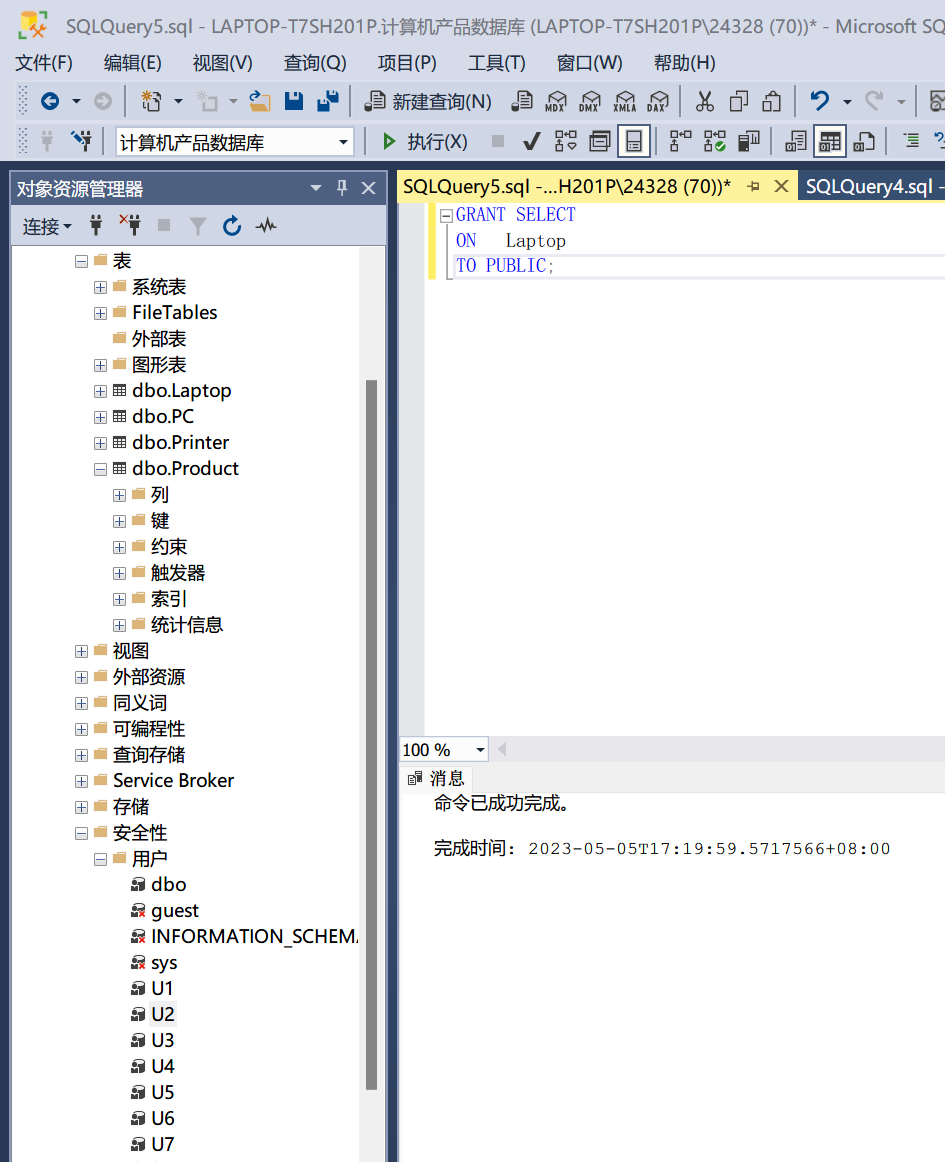


例2：

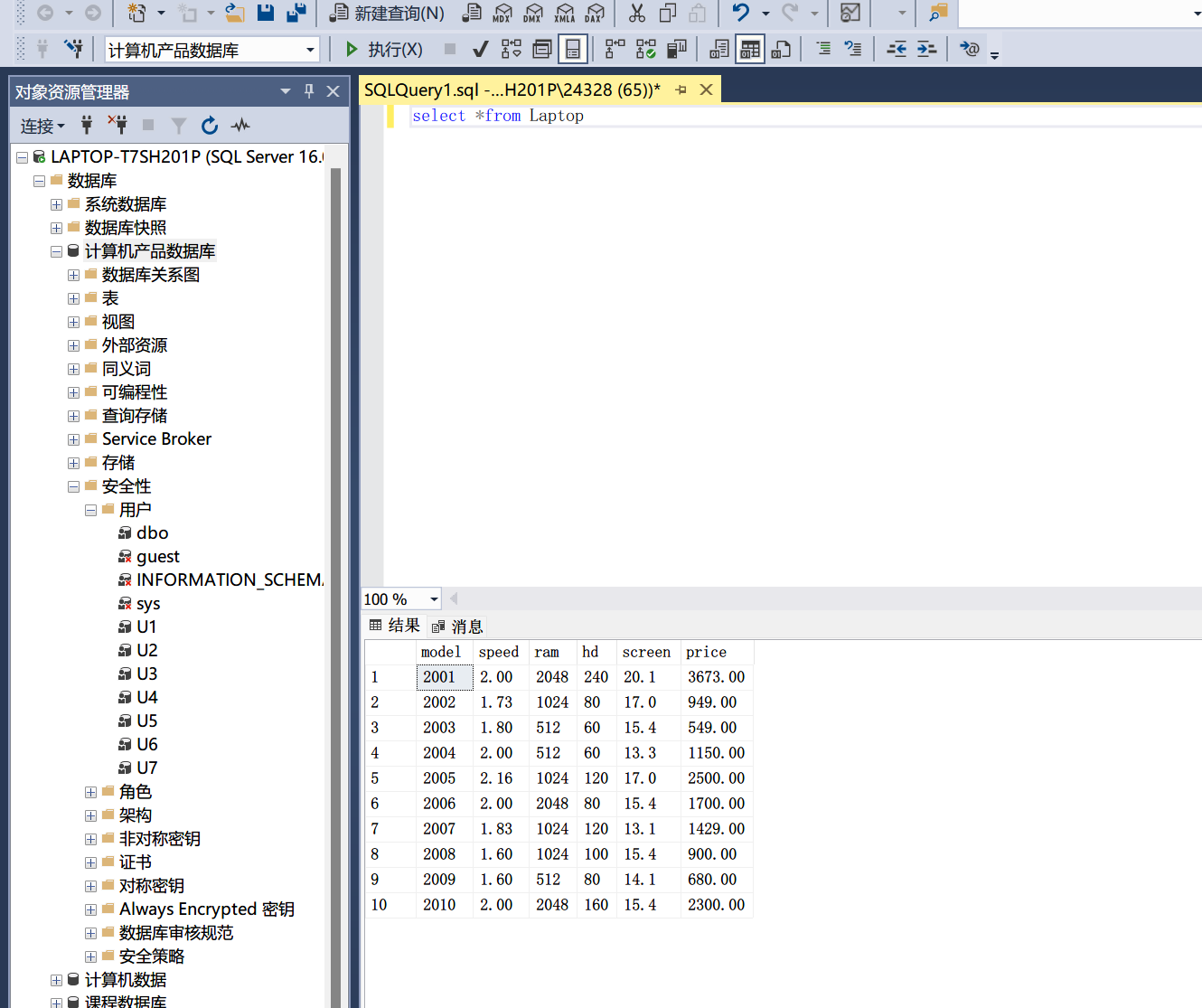
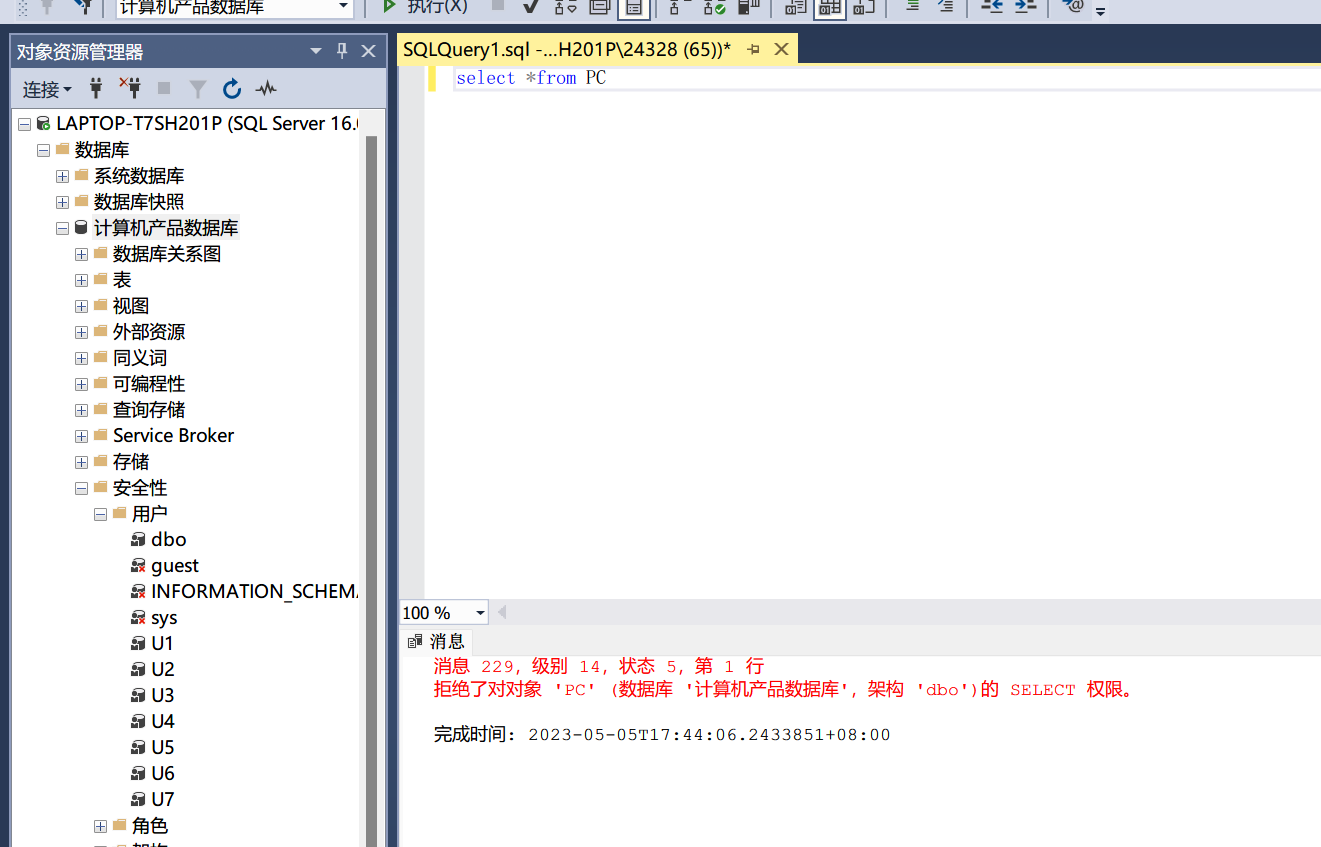




例3：



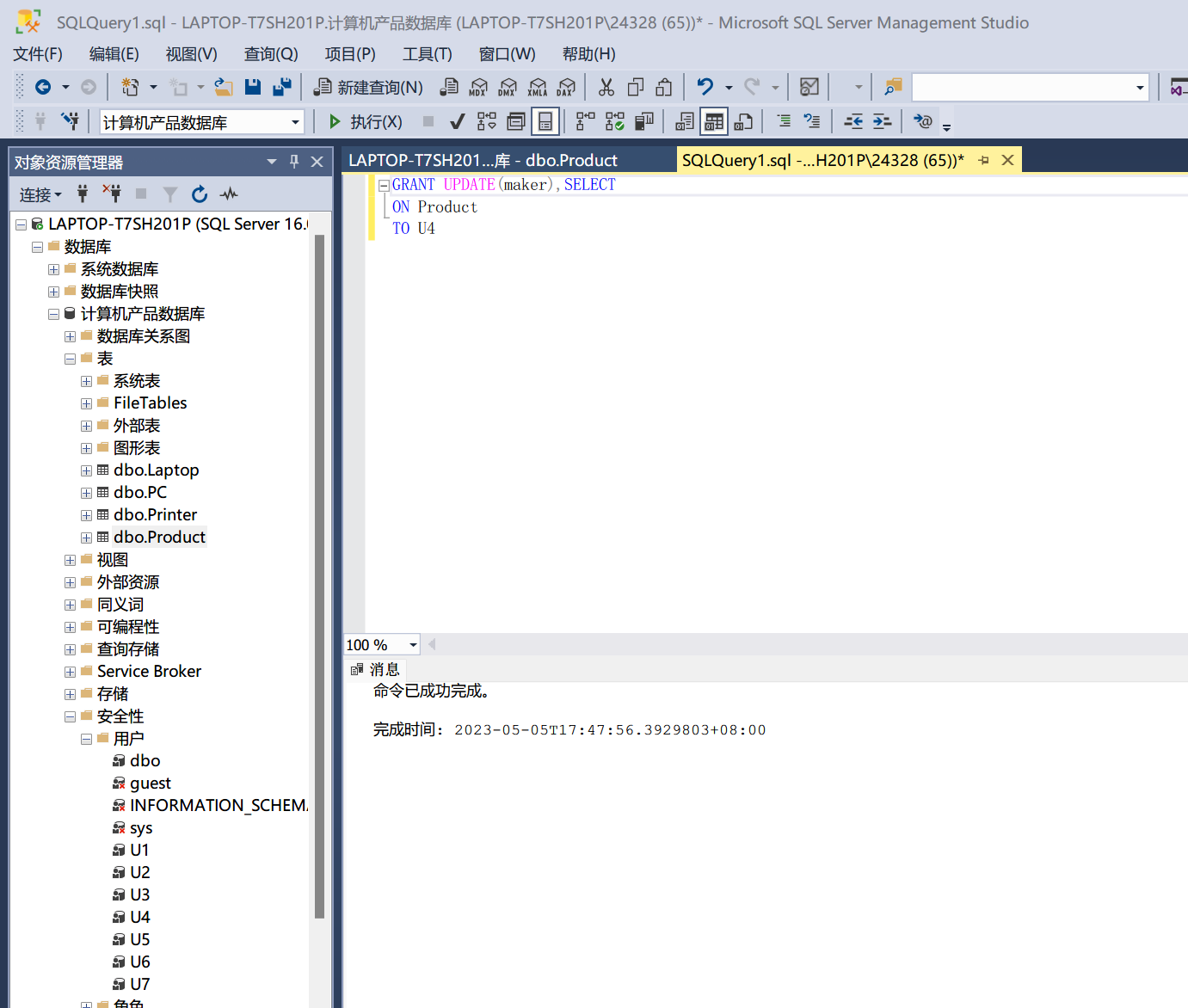
切换到U4



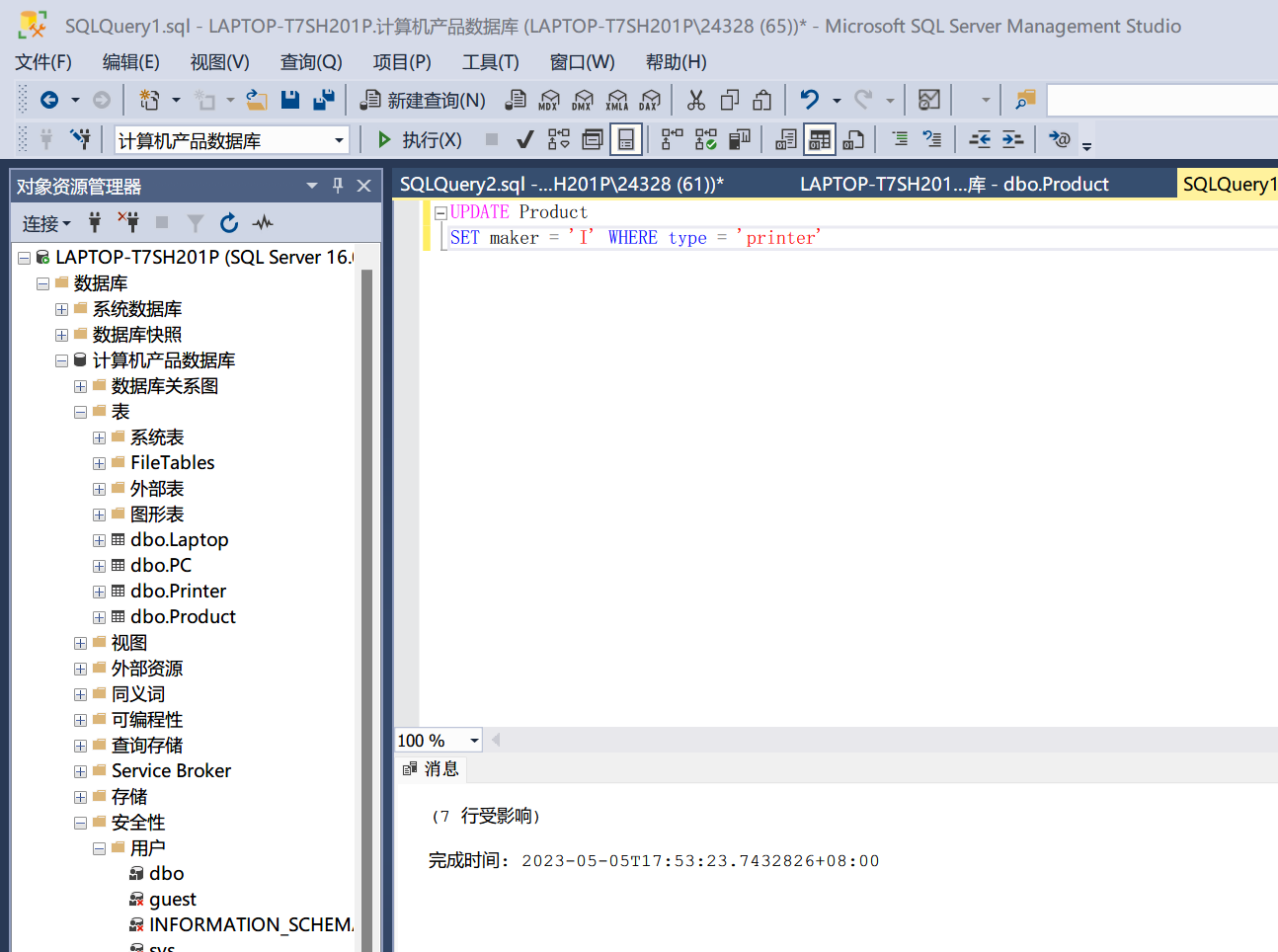
发现PC无法查询，Laptop可查询。

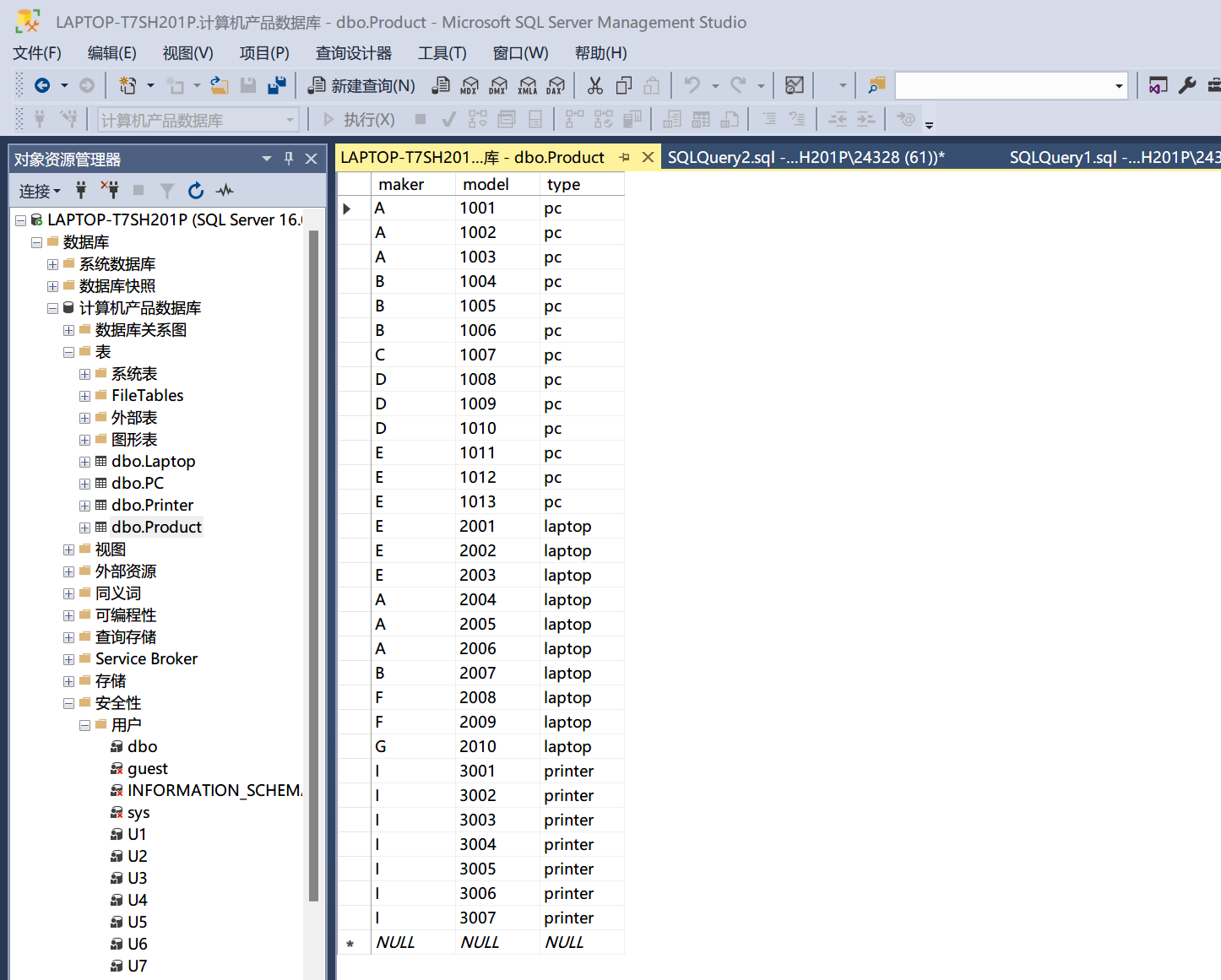
例4：

将把查询Product与修改maker的权限给用户U4

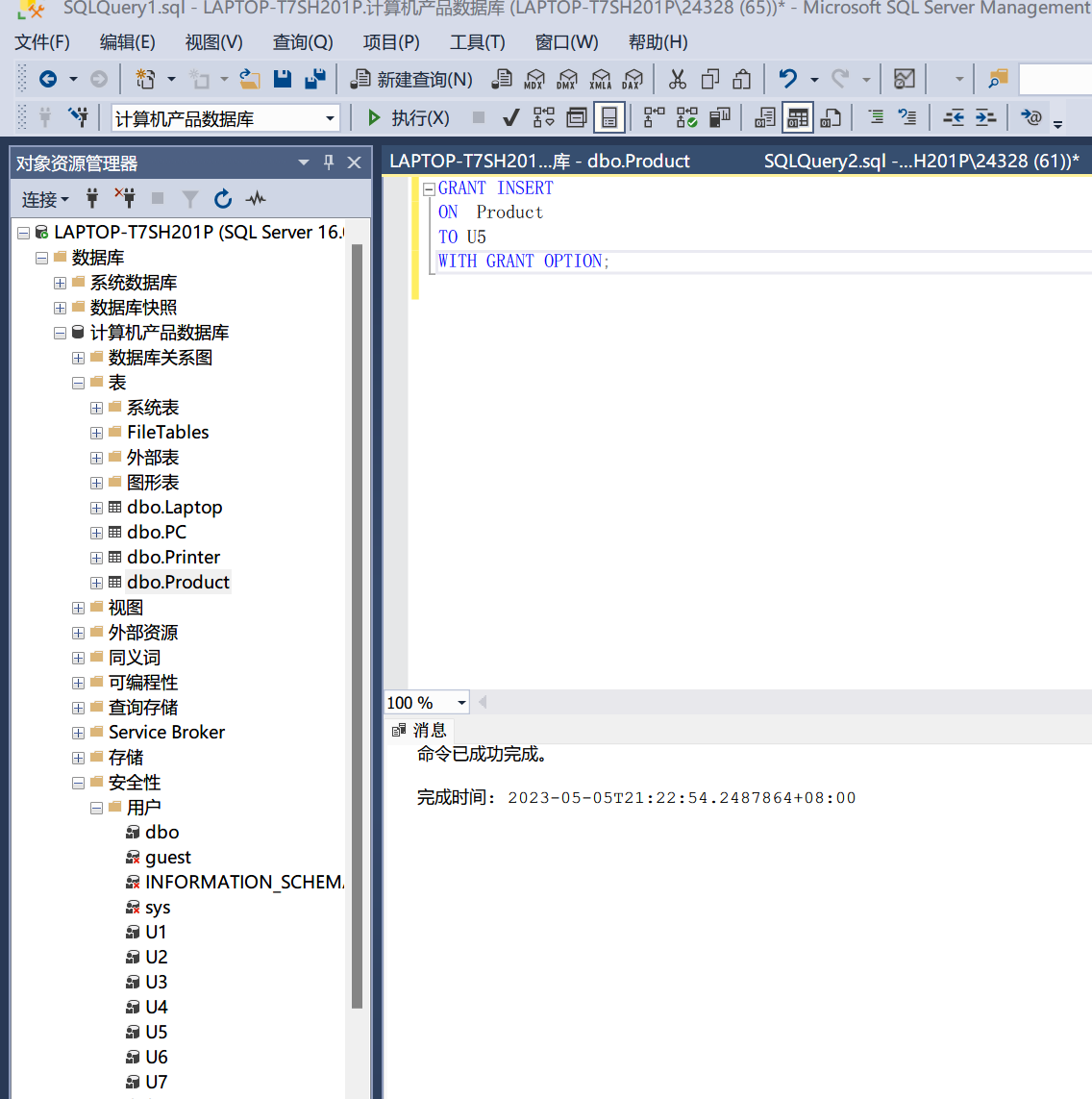


切换到U4 把 Product中 type = printer 的maker改成I



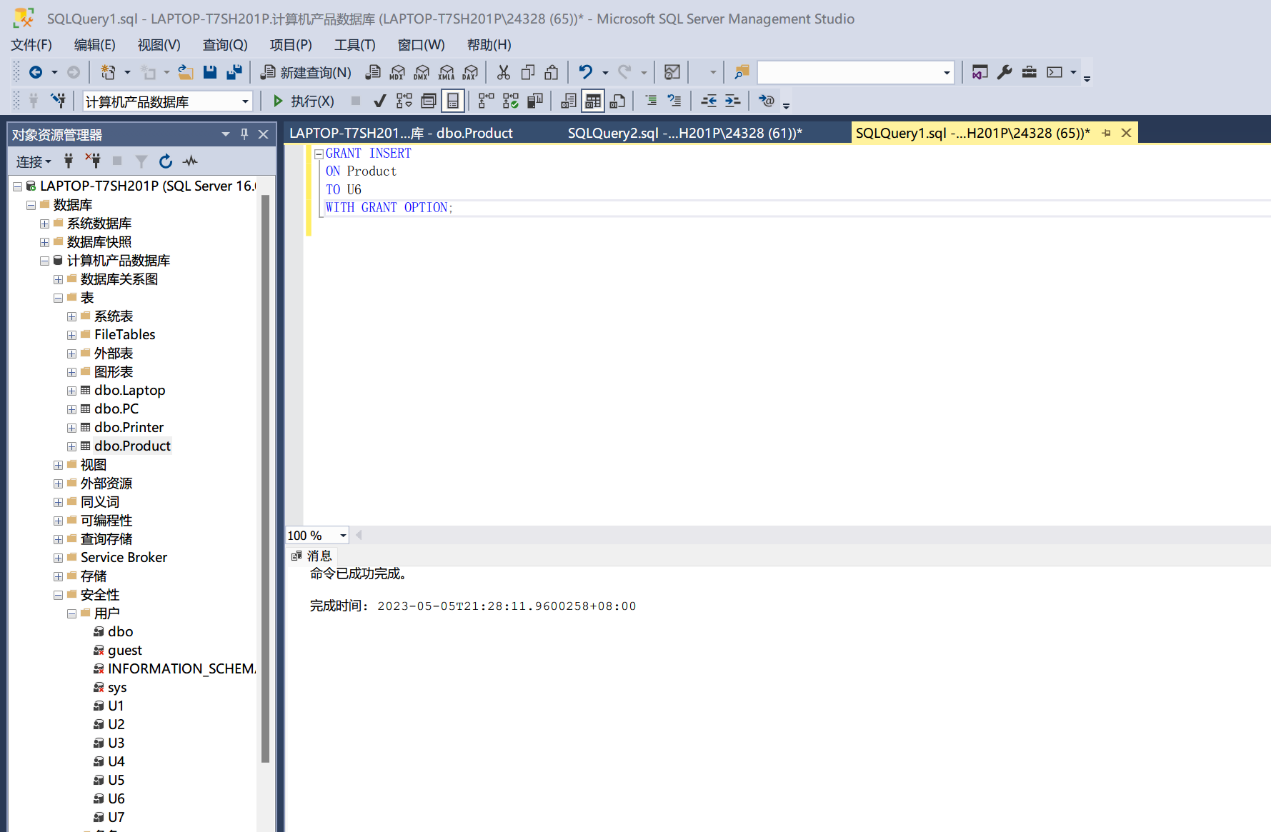


例5：



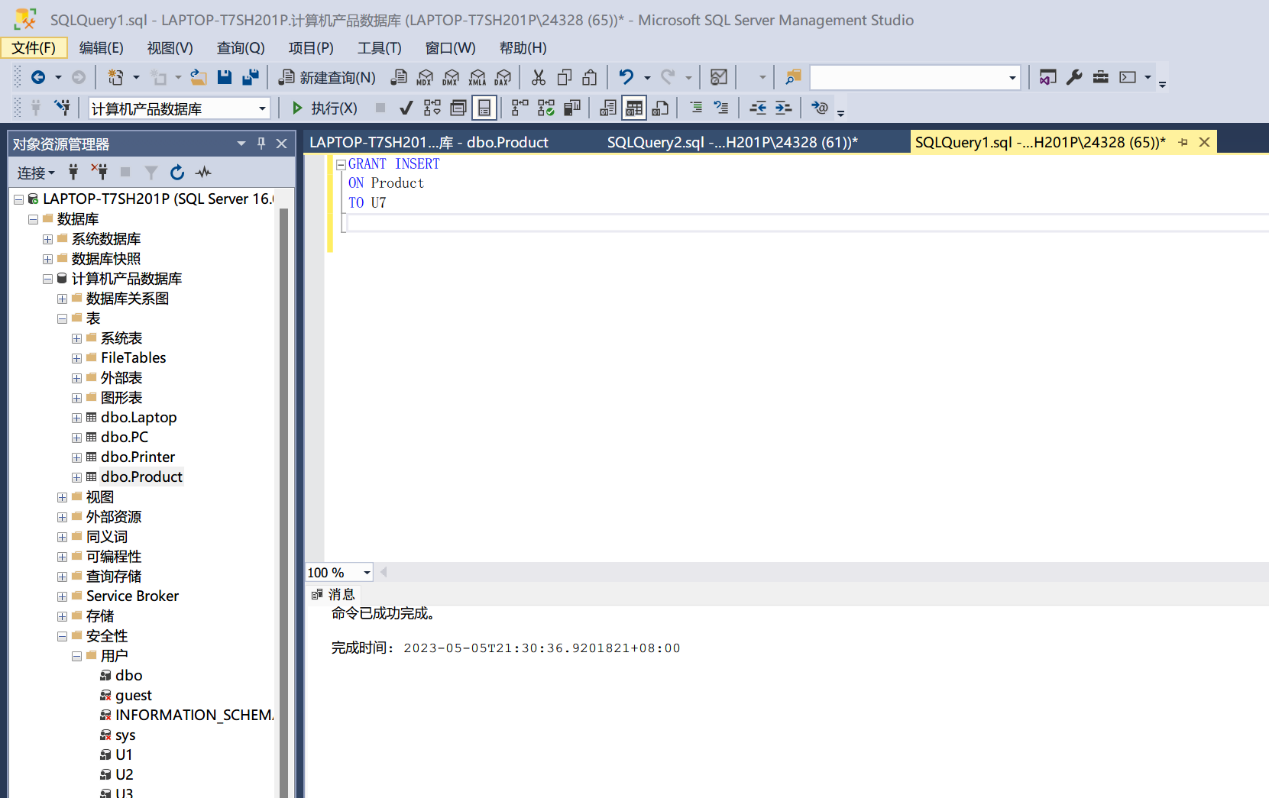
把对表Product的INSERT权限授予U5用户，并允许它将此权限再授予给其他用户

例6：

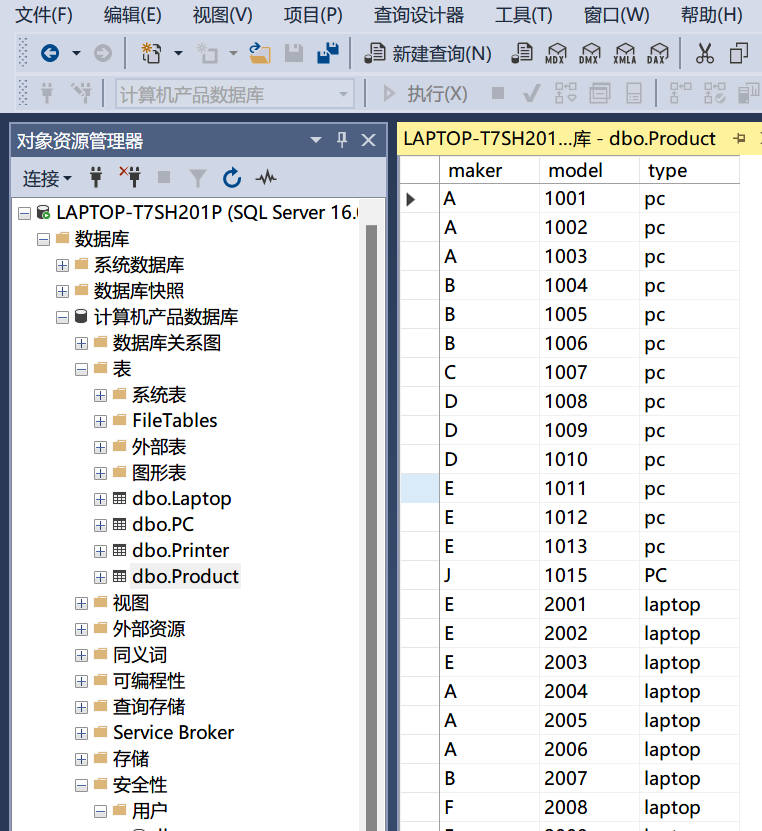
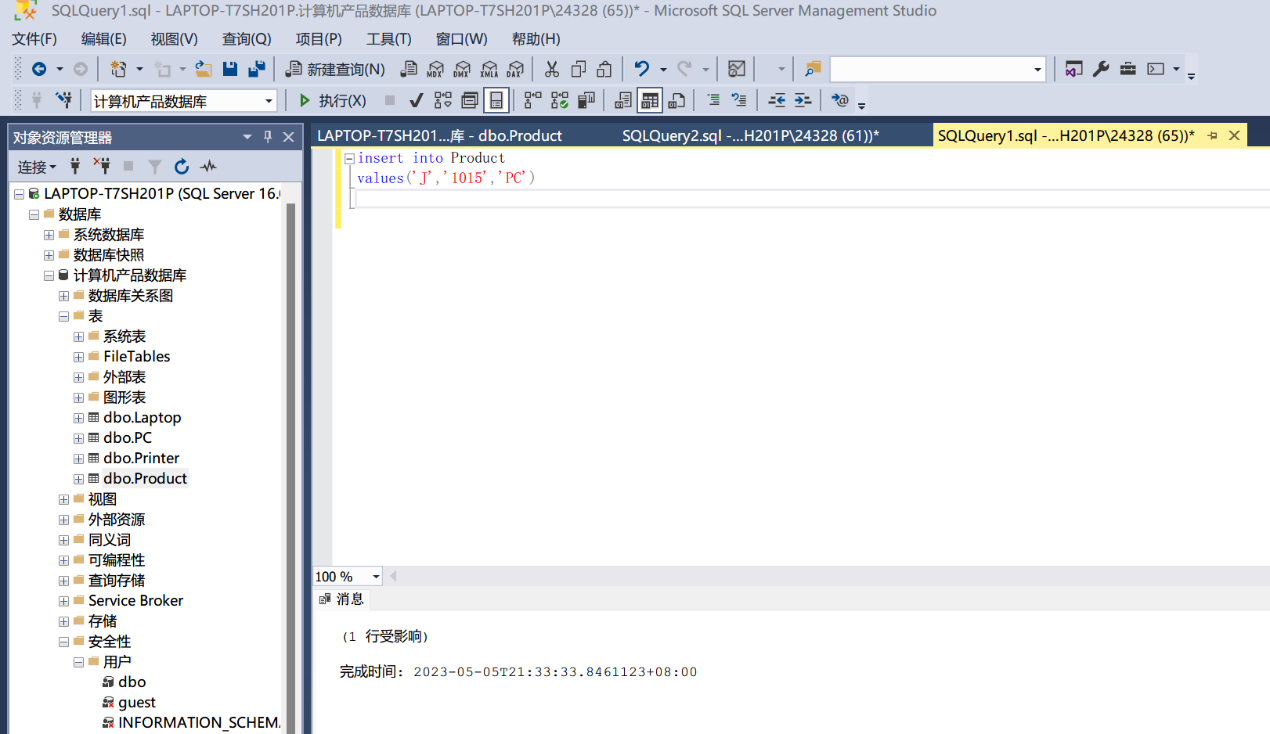


U5将权限授予U6

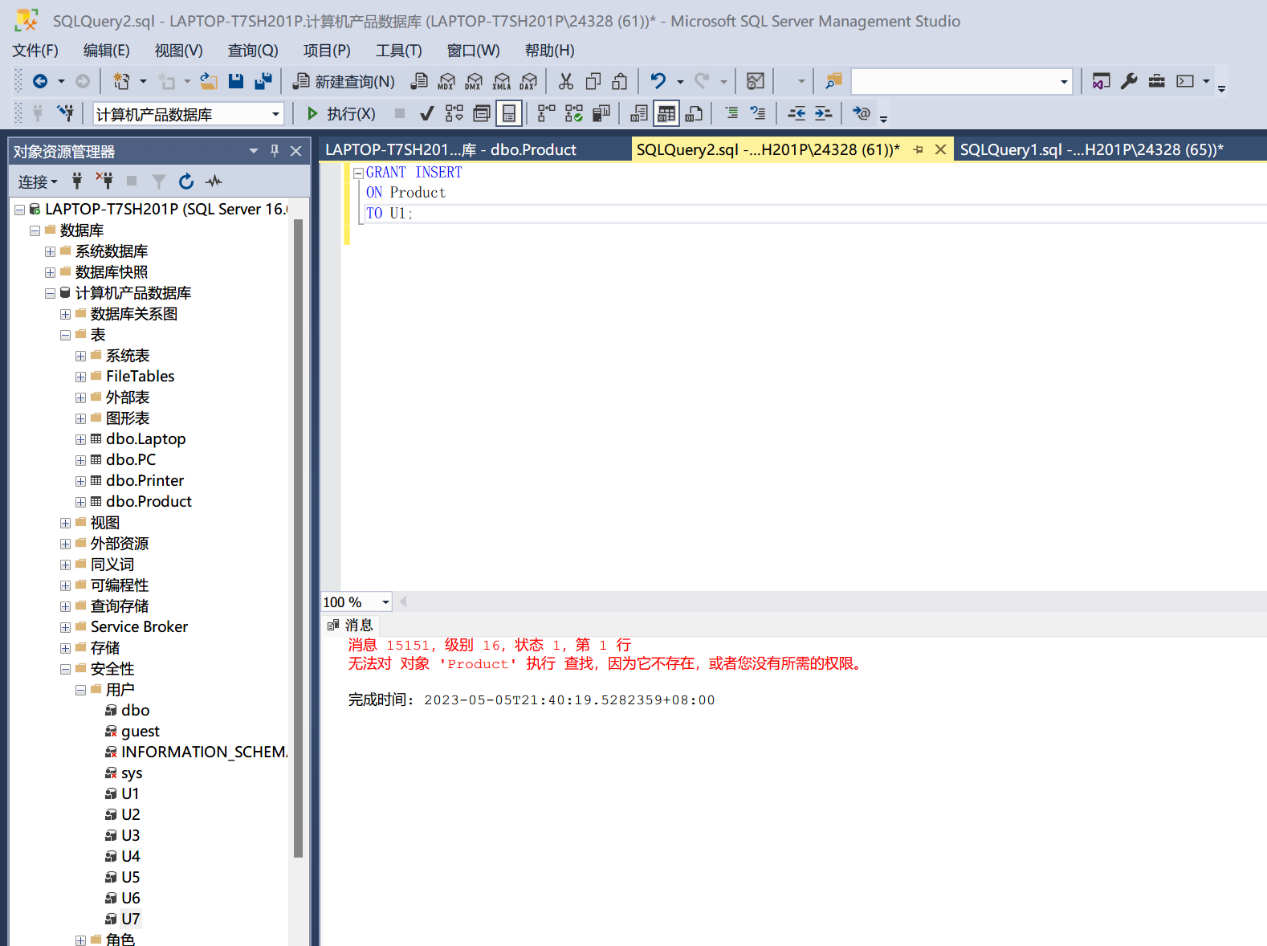
例7：



U5将权限授予U7，但是不允许U7将权限继续授予其他人

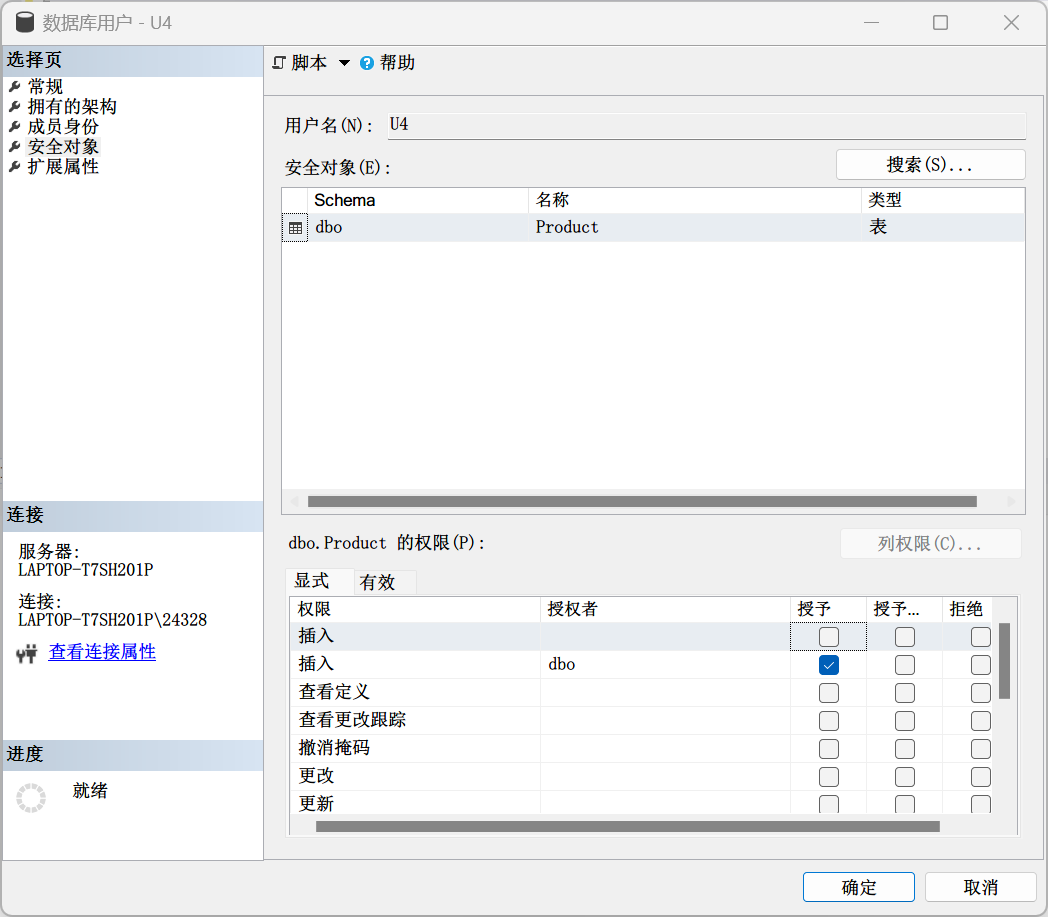
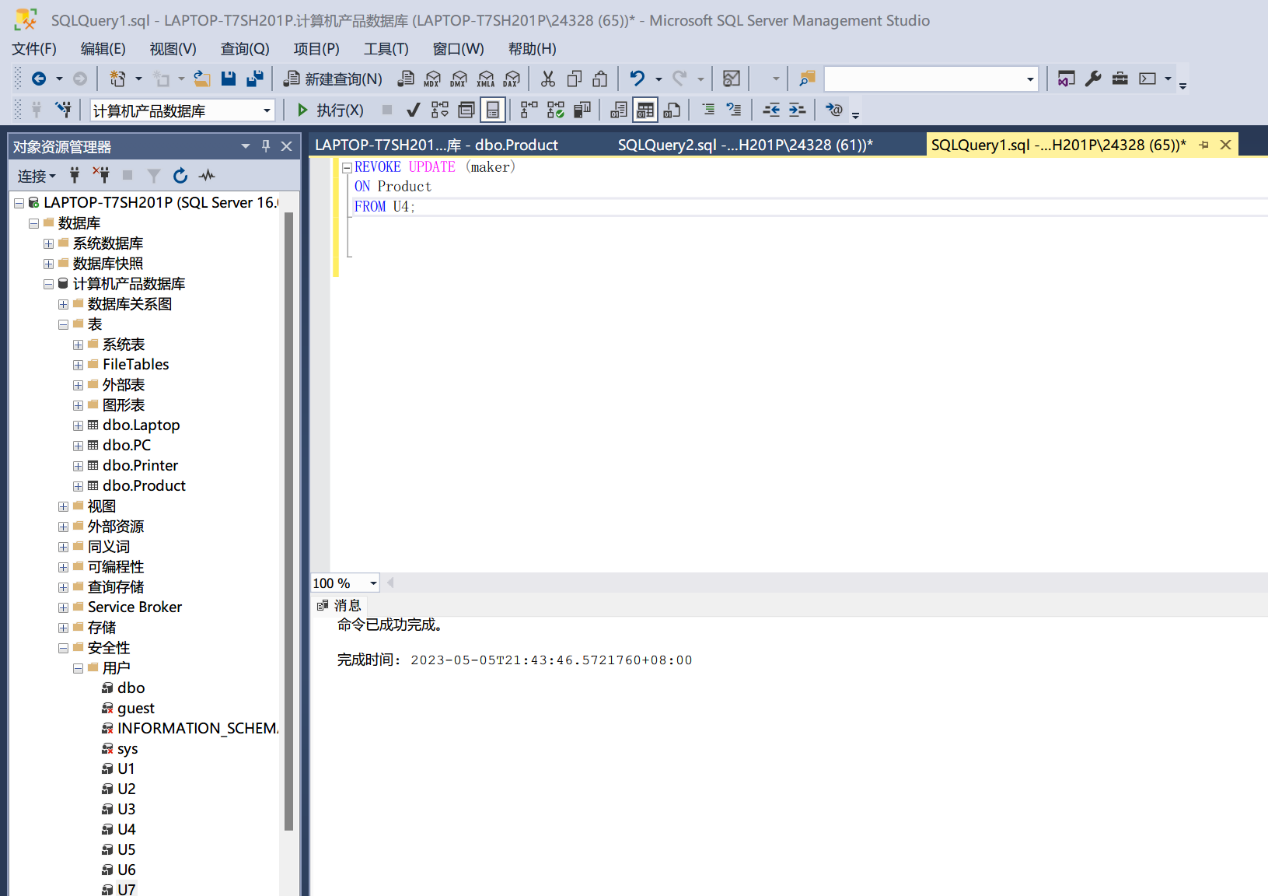


进入U5或者U6,U7发现可以插入，插入'J','1015','PC'。



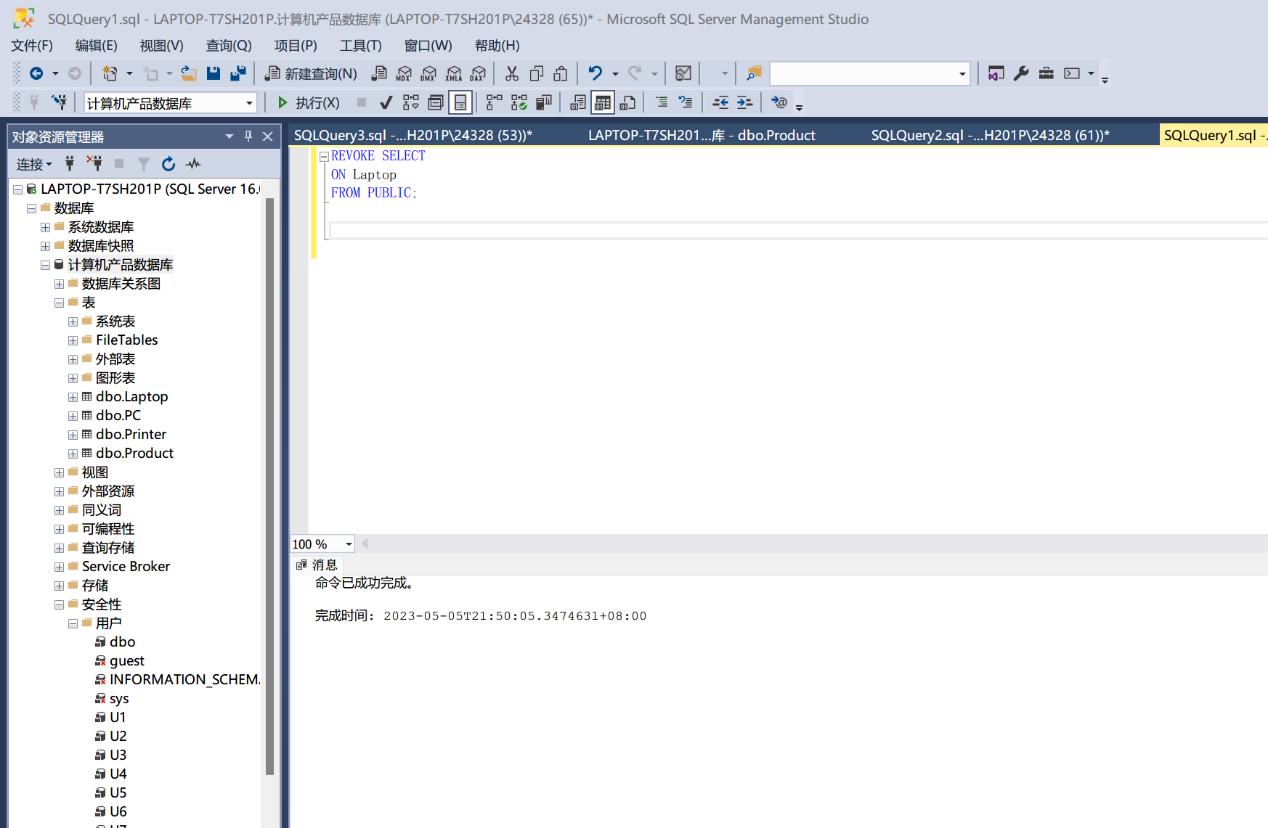
进入U7，因为U7在上一个授权没有获得授权继续给下一个，因此U7无法授权给其他人。

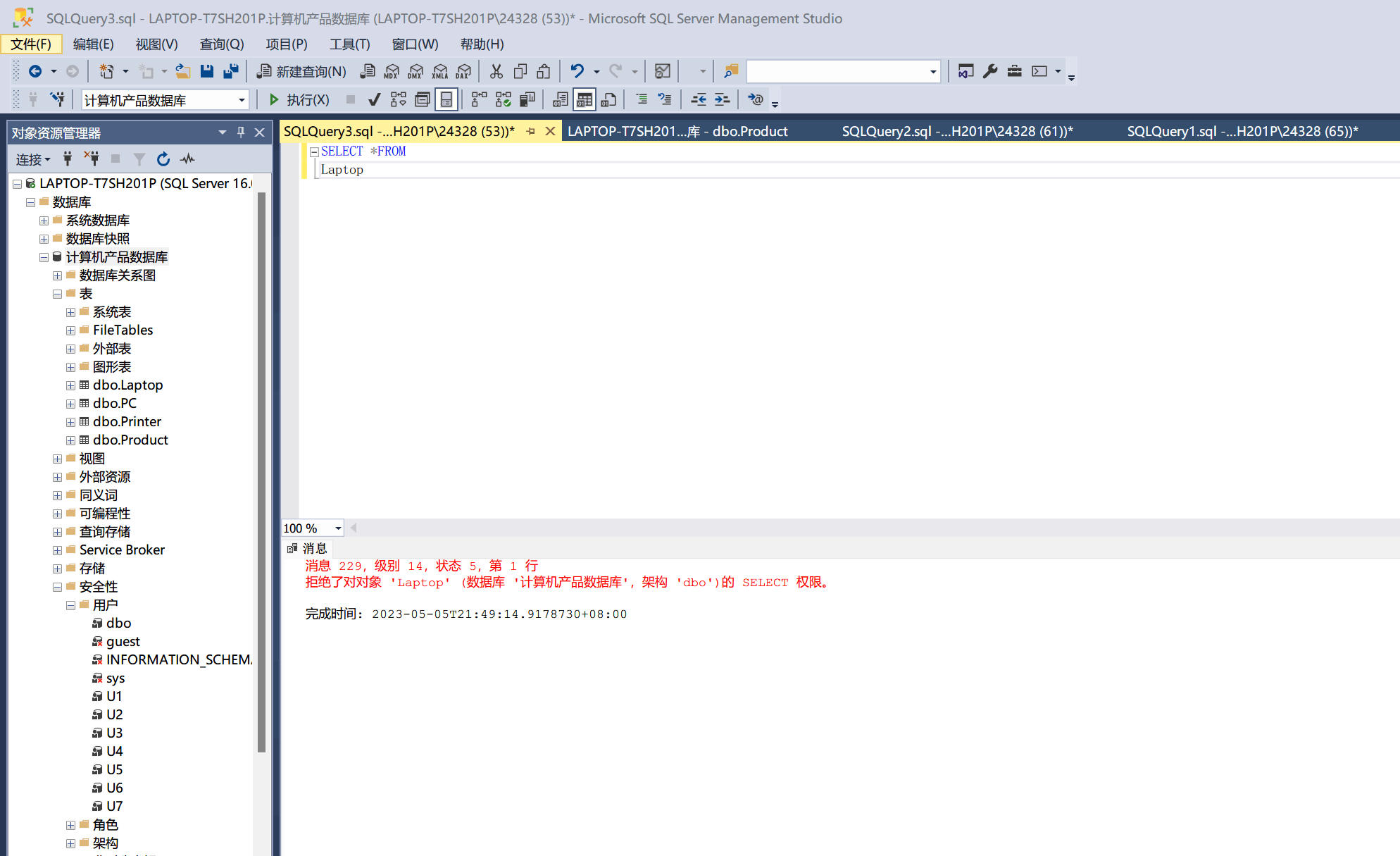
例8：



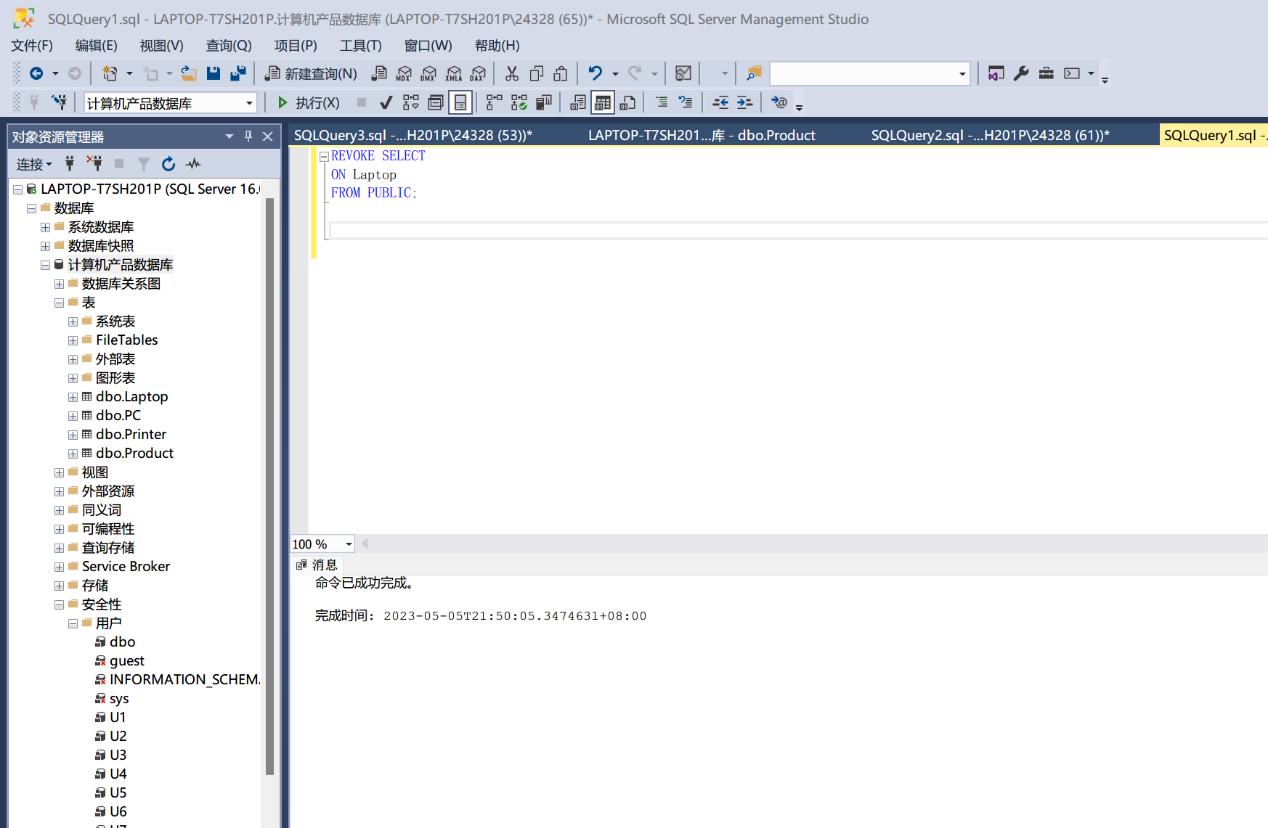
收回U4权限，U4没有更新的权限了。

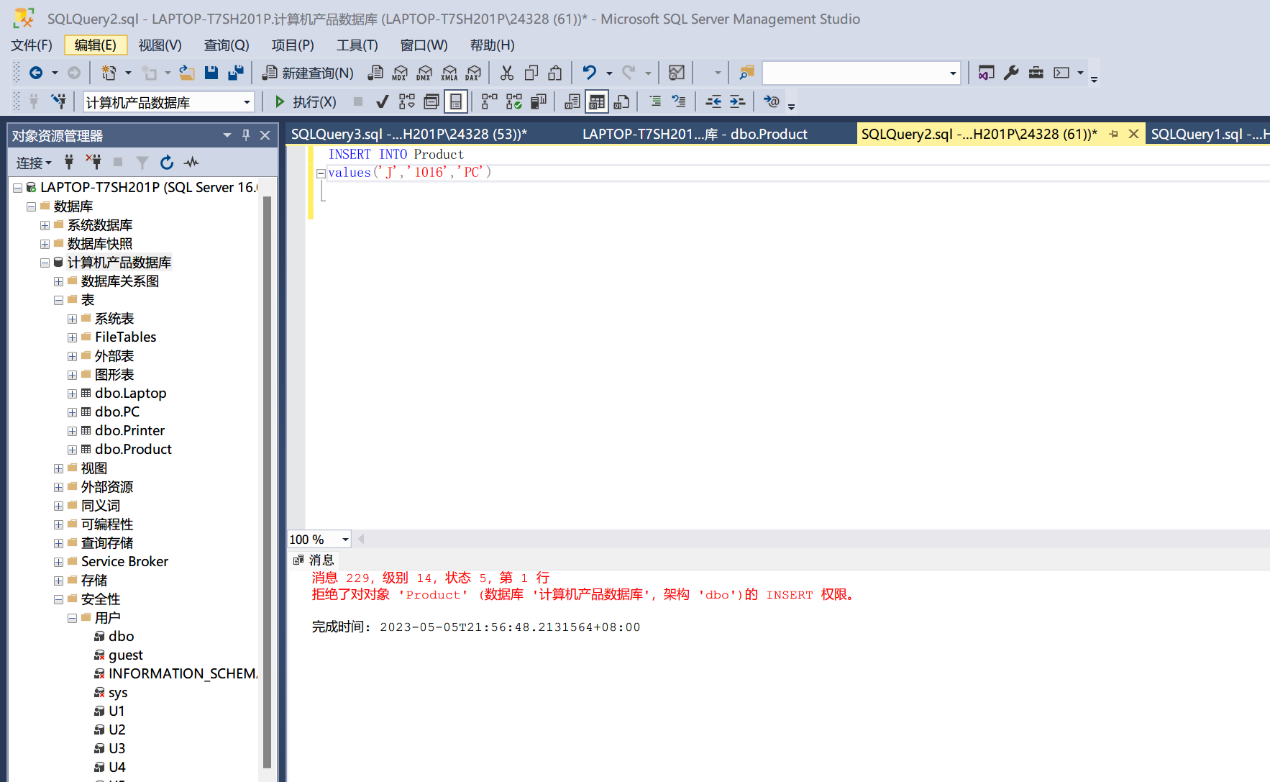
例9：





收回所有用户对表Laptop的查询权限，然后其他的表无法查询Laptop。

例10：



收回U5的INSERT对Product的权限，进入U7发现U7的权限也被收回了，因为使用了CASCADE把授予的权限也收回。

### 2.完整性部分:

使用SQL对数据进行完整性控制，并用实验证实，当操作违反了完整性约束条件时，系统是如何处理的

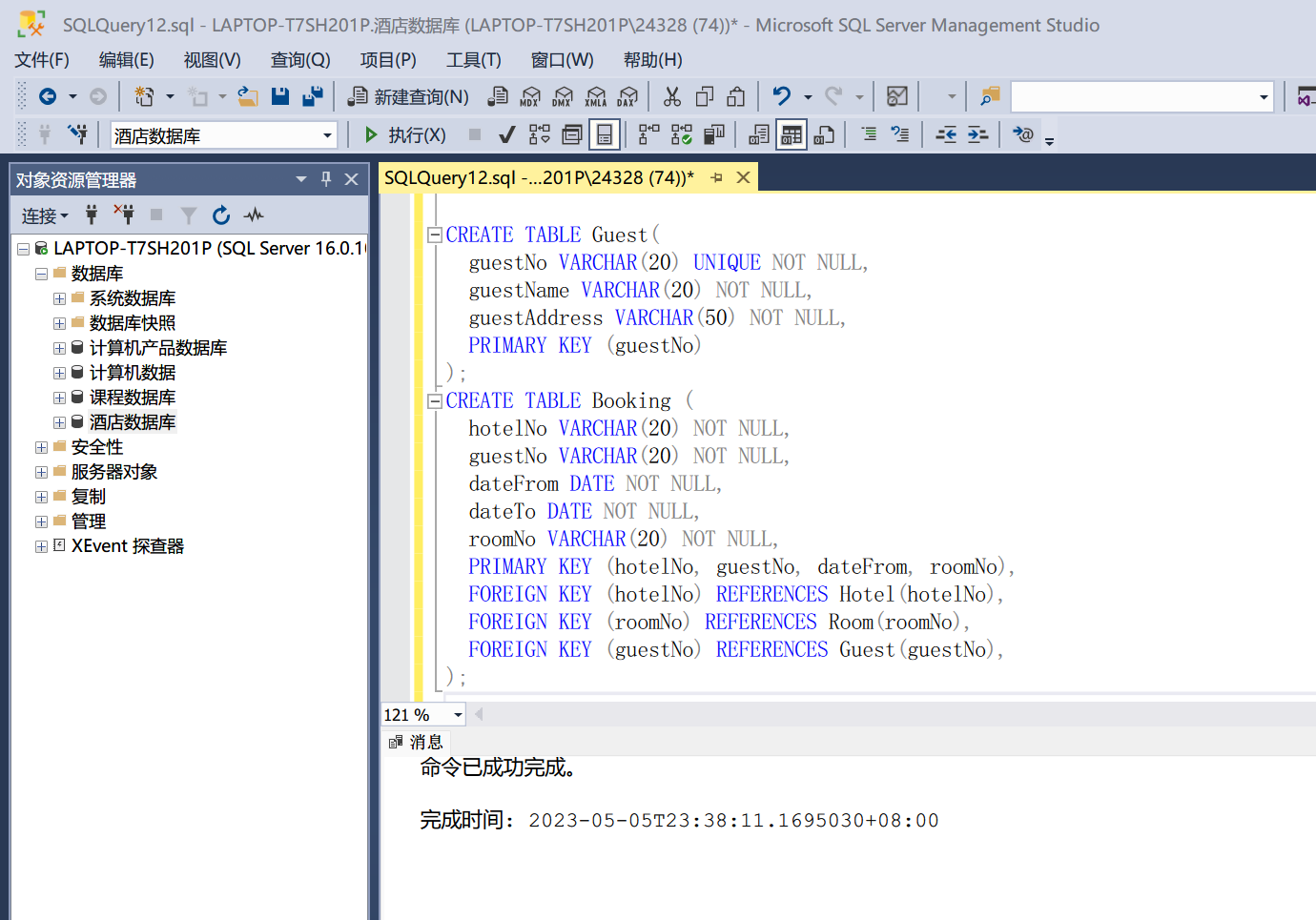
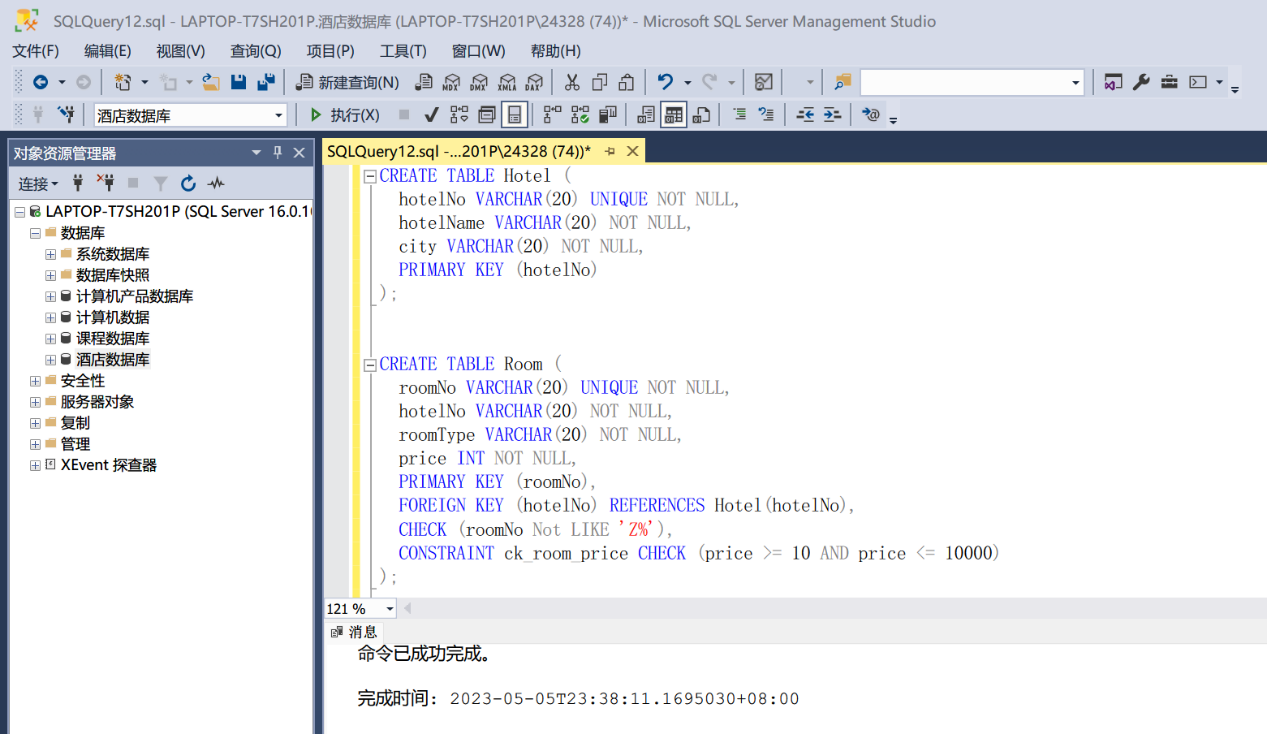
实体完整性(仿照[例1]、[例2] ) x1

参照完整性(仿照[例3]) x4

用户定义完整性(仿照[例5]、 [例6]) x1

CHECK短语(仿照[例7]或 [例8] 、[例9]) x1

CONSTRAINT子句(仿照[例10]、 [例13]) x1



代码如下：

CREATE TABLE Hotel (

hotelNo VARCHAR(20) UNIQUE NOT NULL,

hotelName VARCHAR(20) NOT NULL,

city VARCHAR(20) NOT NULL,

PRIMARY KEY (hotelNo)

);

CREATE TABLE Room (

roomNo VARCHAR(20) UNIQUE NOT NULL,

hotelNo VARCHAR(20) NOT NULL,

roomType VARCHAR(20) NOT NULL,

price INT NOT NULL,

PRIMARY KEY (roomNo),

FOREIGN KEY (hotelNo) REFERENCES Hotel(hotelNo),

CHECK (roomNo Not LIKE 'Z%'),

CONSTRAINT ck\_room\_price CHECK (price >= 10 AND price <= 10000)

);

CREATE TABLE Guest(

guestNo VARCHAR(20) UNIQUE NOT NULL,

guestName VARCHAR(20) NOT NULL,

guestAddress VARCHAR(50) NOT NULL,

PRIMARY KEY (guestNo)

);

CREATE TABLE Booking (

hotelNo VARCHAR(20) NOT NULL,

guestNo VARCHAR(20) NOT NULL,

dateFrom DATE NOT NULL,

dateTo DATE NOT NULL,

roomNo VARCHAR(20) NOT NULL,

PRIMARY KEY (hotelNo, guestNo, dateFrom, roomNo),

FOREIGN KEY (hotelNo) REFERENCES Hotel(hotelNo),

FOREIGN KEY (roomNo) REFERENCES Room(roomNo),

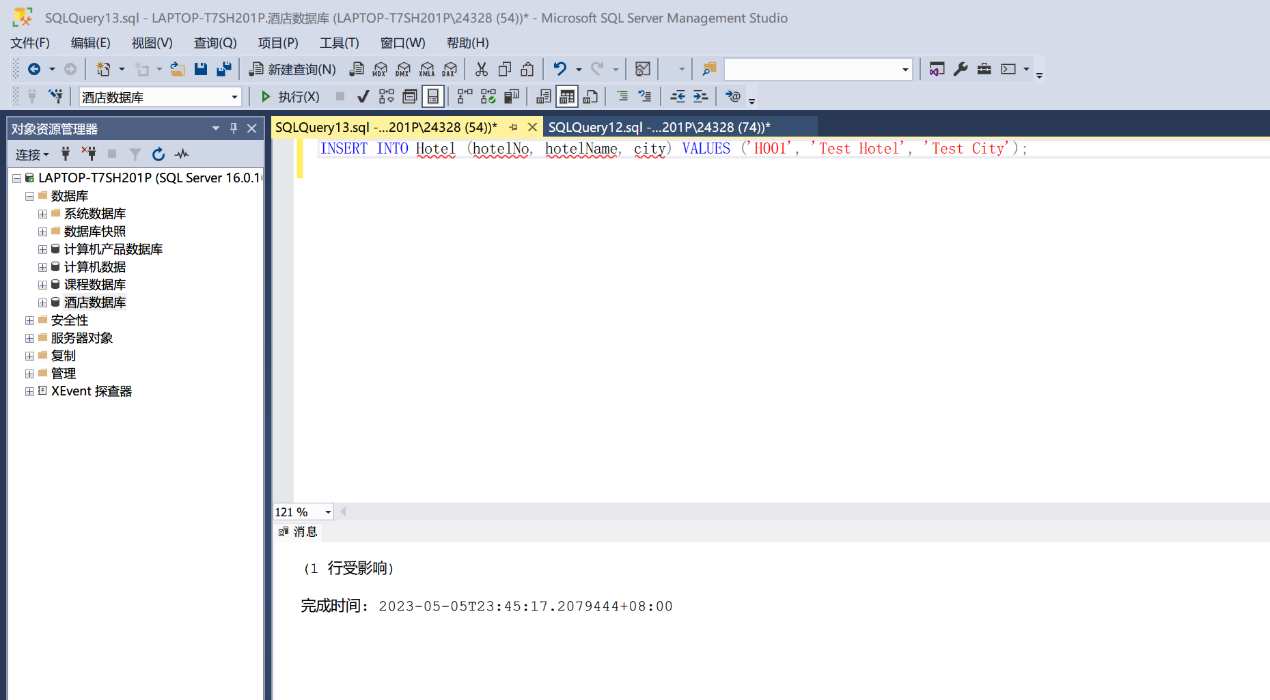
FOREIGN KEY (guestNo) REFERENCES Guest(guestNo),

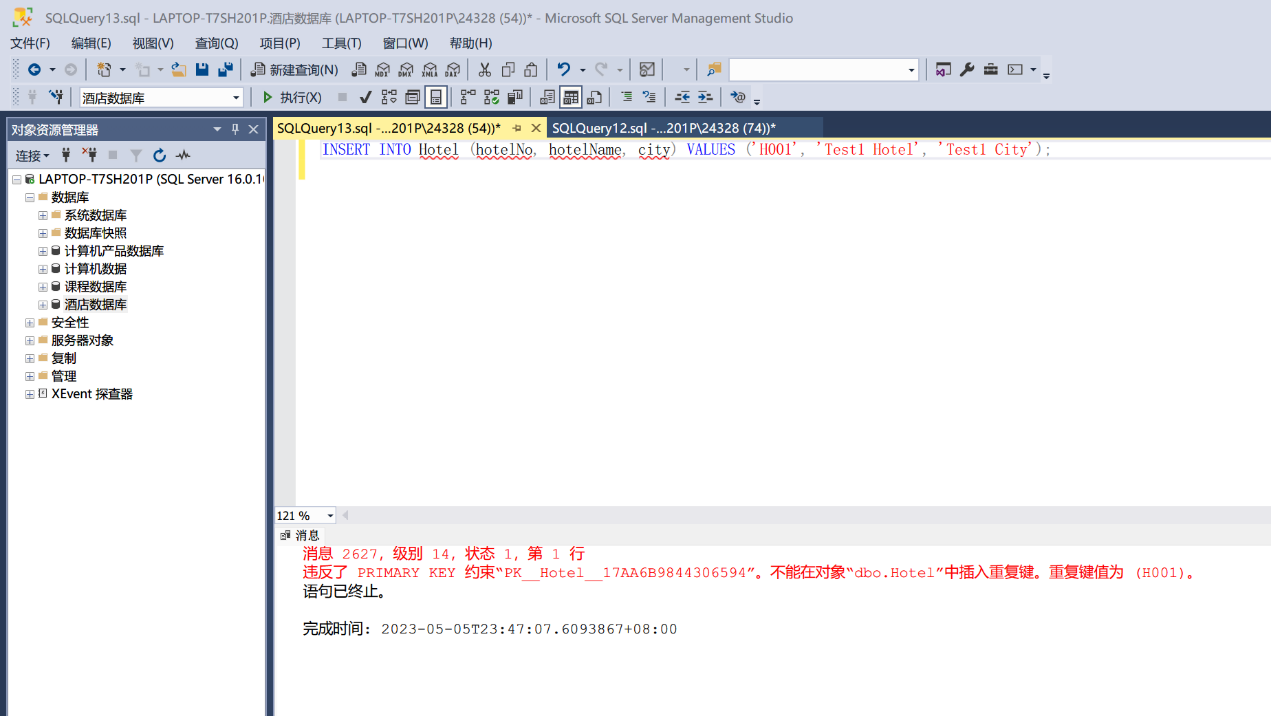
);

创建酒店数据库，用PRIMARY定义码实现实体完整性，用如FOREIGN KEY (guestNo) REFERENCES Guest(guestNo)实现参照完整性，对一些值不允许取空，用NOT NULL以及对一些列值取唯一，用UNIQUE NOT NULL,实现参照完整性。用CHECK (roomNo Not LIKE 'Z%'),确保roomNo不以Z开头实现CHECK语句。再用CONSTRAINT ck\_room\_price CHECK (price >= 10 AND price <= 10000)实现CONSTRAINT语句。使得可以灵活增加，删除一个完整性约束条件。

#### 验证实体完整性

- 尝试在 Hotel 表中插入一个具有相同主键的记录：

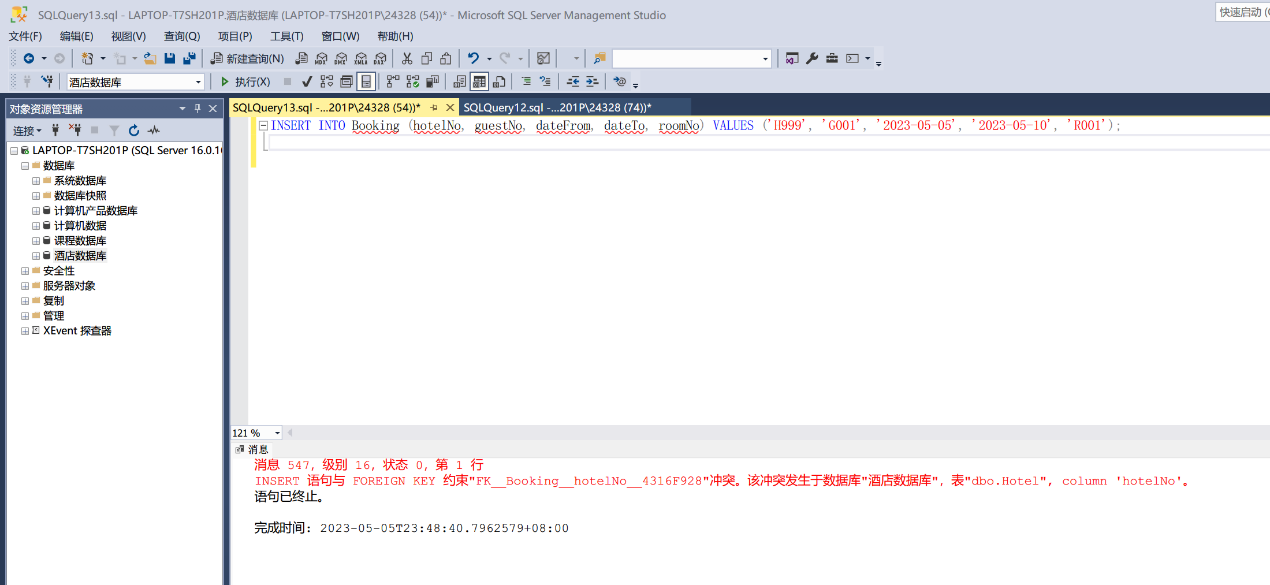




先在表中插入一个Hotel中主键为H001的数据，再插入一个主键相同的数据，显示报错。

验证参照完整性

- 尝试在 Booking 表中插入一个 hotelNo 不存在的记录：



结果显示报错，因为我们试图在 Booking 表中插入一个 hotelNo 不存在的记录，由于 hotelNo 是一个外键，它必须参照 Hotel 表中的一个已存在的记录。

#### 验证用户定义完整性

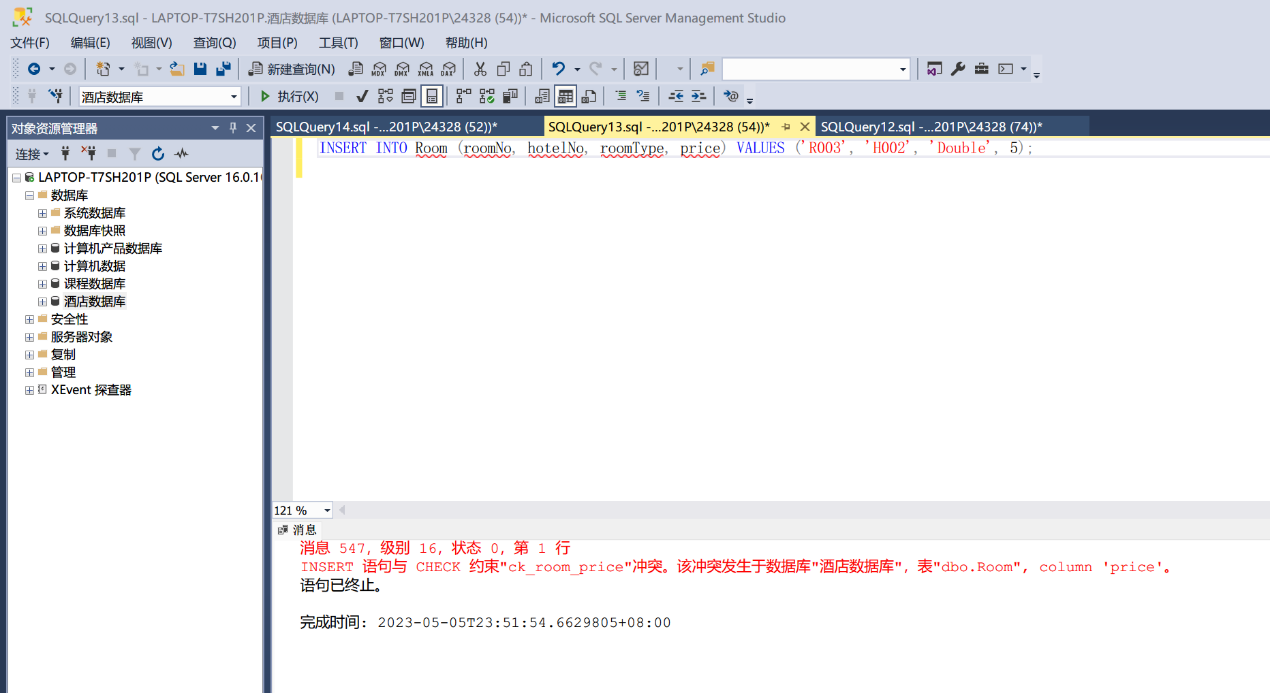
- 尝试在 Room 表中插入一个房间编号以 "Z" 开头的记录：



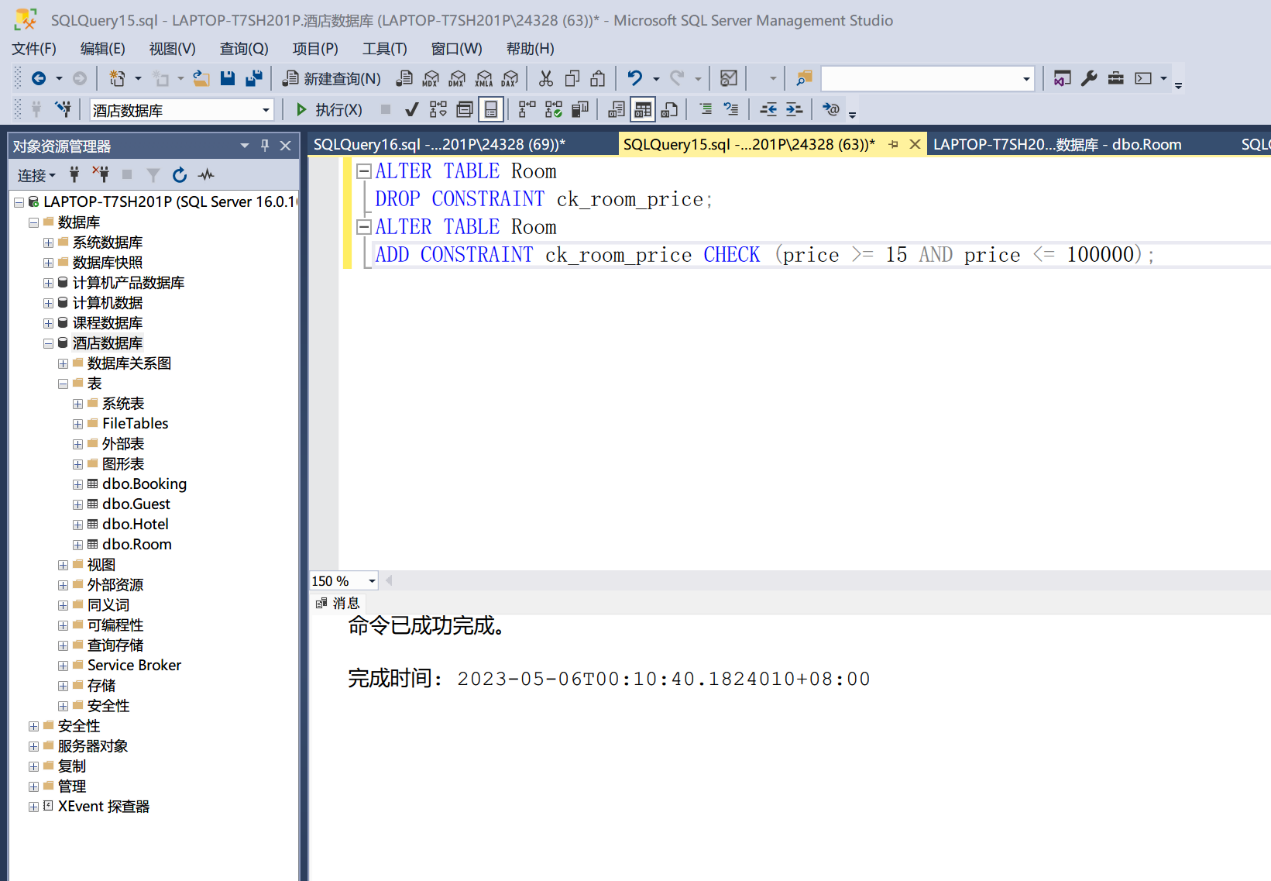
报错，因为我们在 Room 表中使用 CHECK 约束限制了 roomNo 不以 "Z" 开头，所以我们不能插入一个以 "Z" 开头的记录

验证 CHECK 短语与CONSTRAINT子句

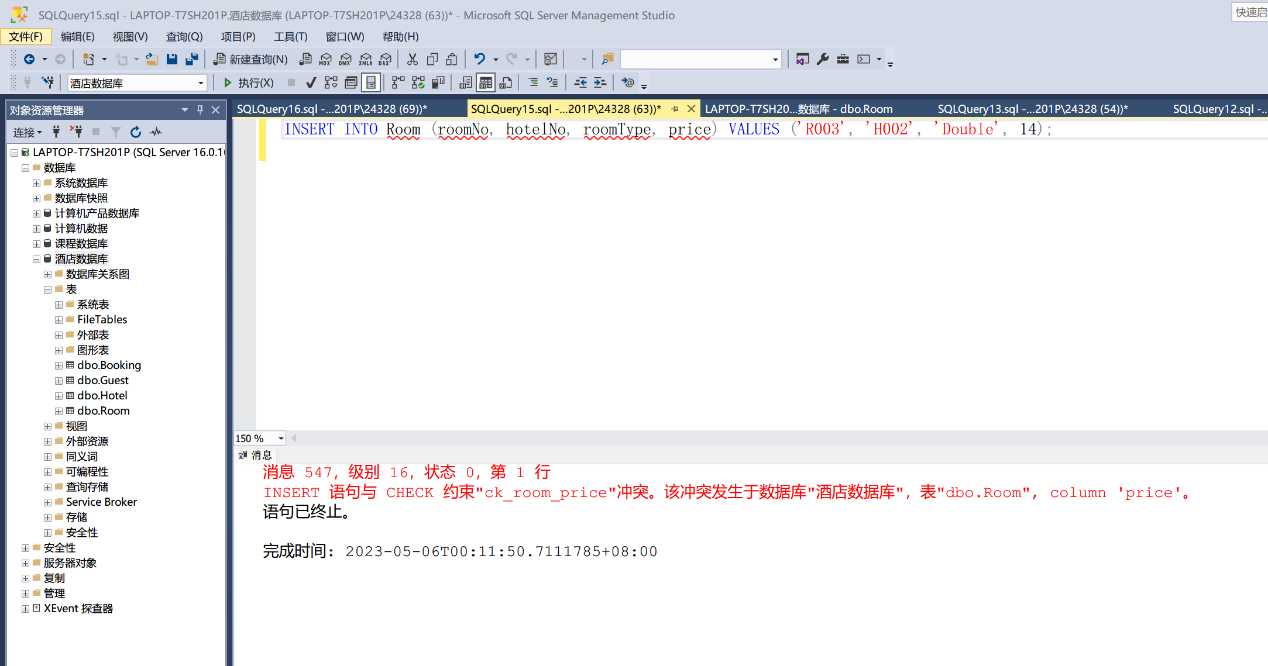
- 尝试在 Room 表中插入一个价格小于 10 的记录：



报错，因为我们在 Room 表中使用 CHECK 约束限制了 price 的范围为 10-10000，所以我们不能插入一个价格小于 10 的记录。



#### 修改CONSTRAINT约束

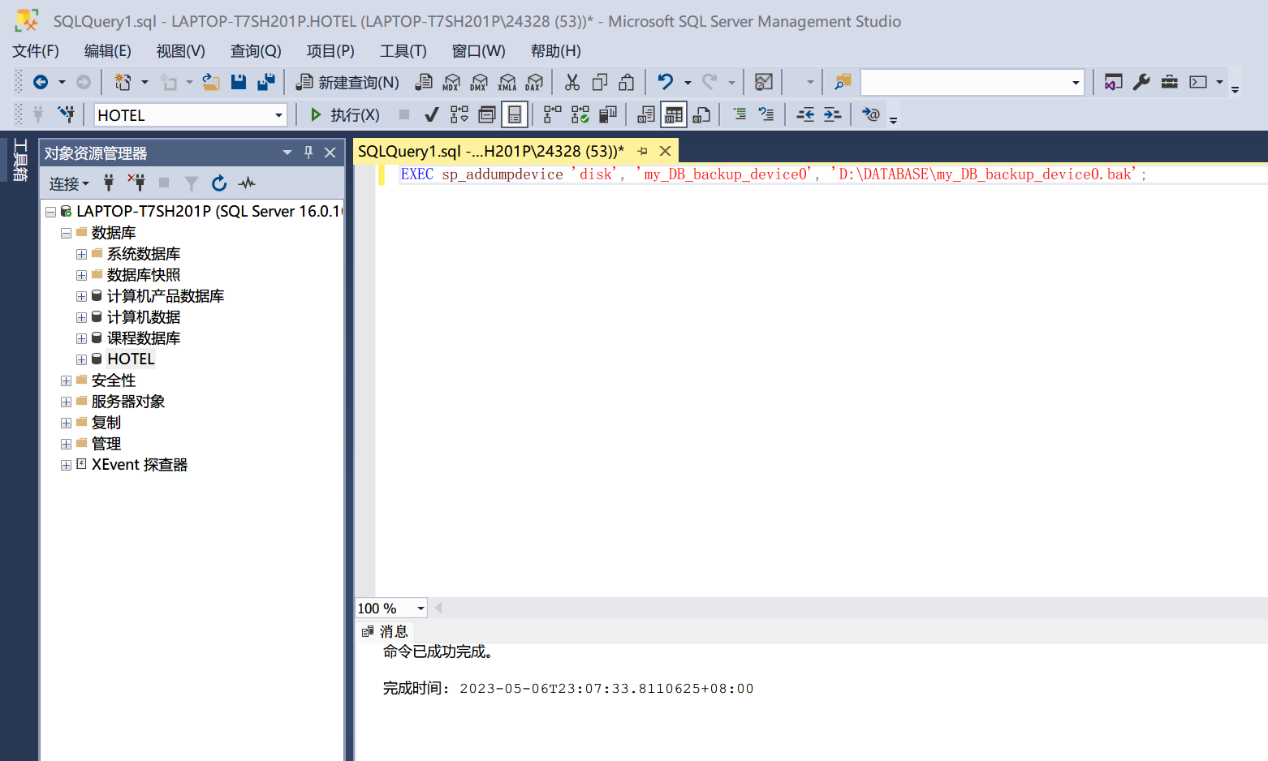


此时插入price>10也没有用，因为约束以及改为了>=15

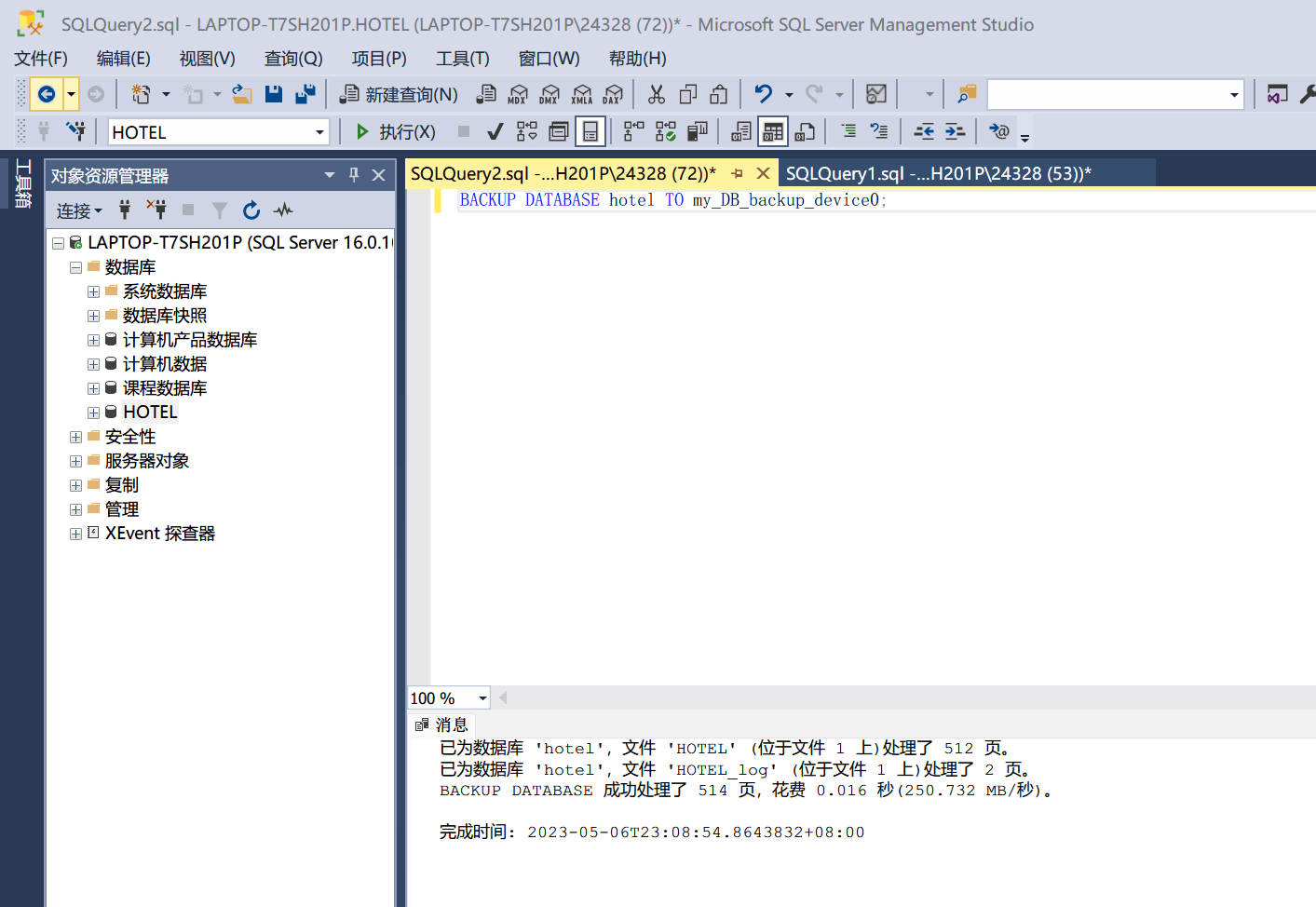
### 3.备份与恢复:

用Transact-SQL完成以下操作

1. 创建一个备份设备。

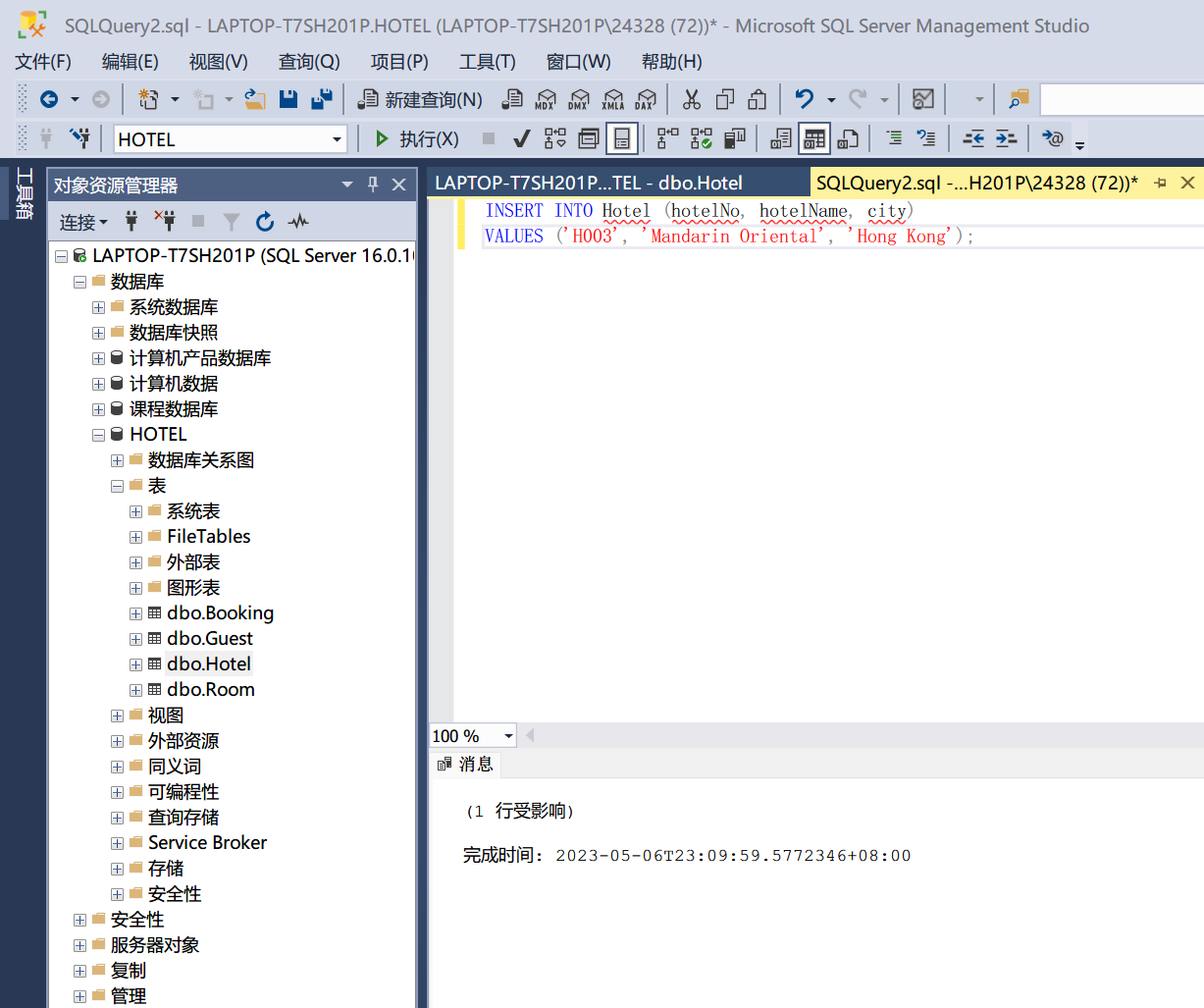


1. 完整备份实验数据库到备份设备。

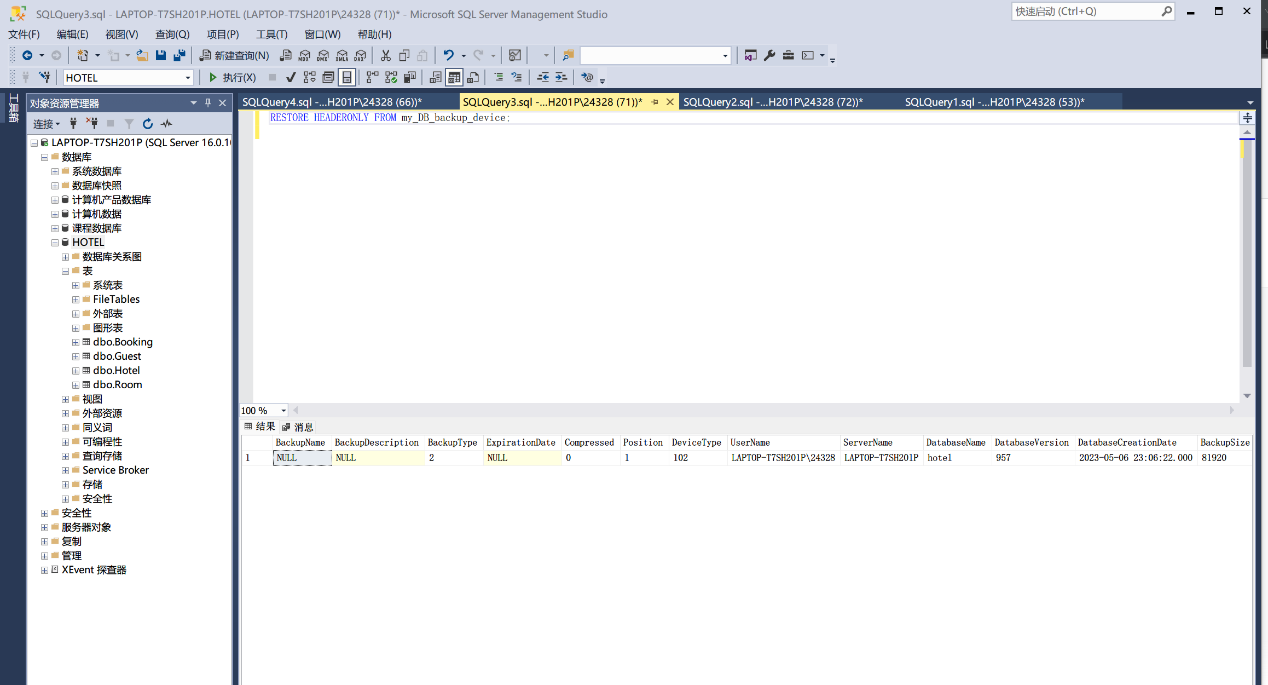
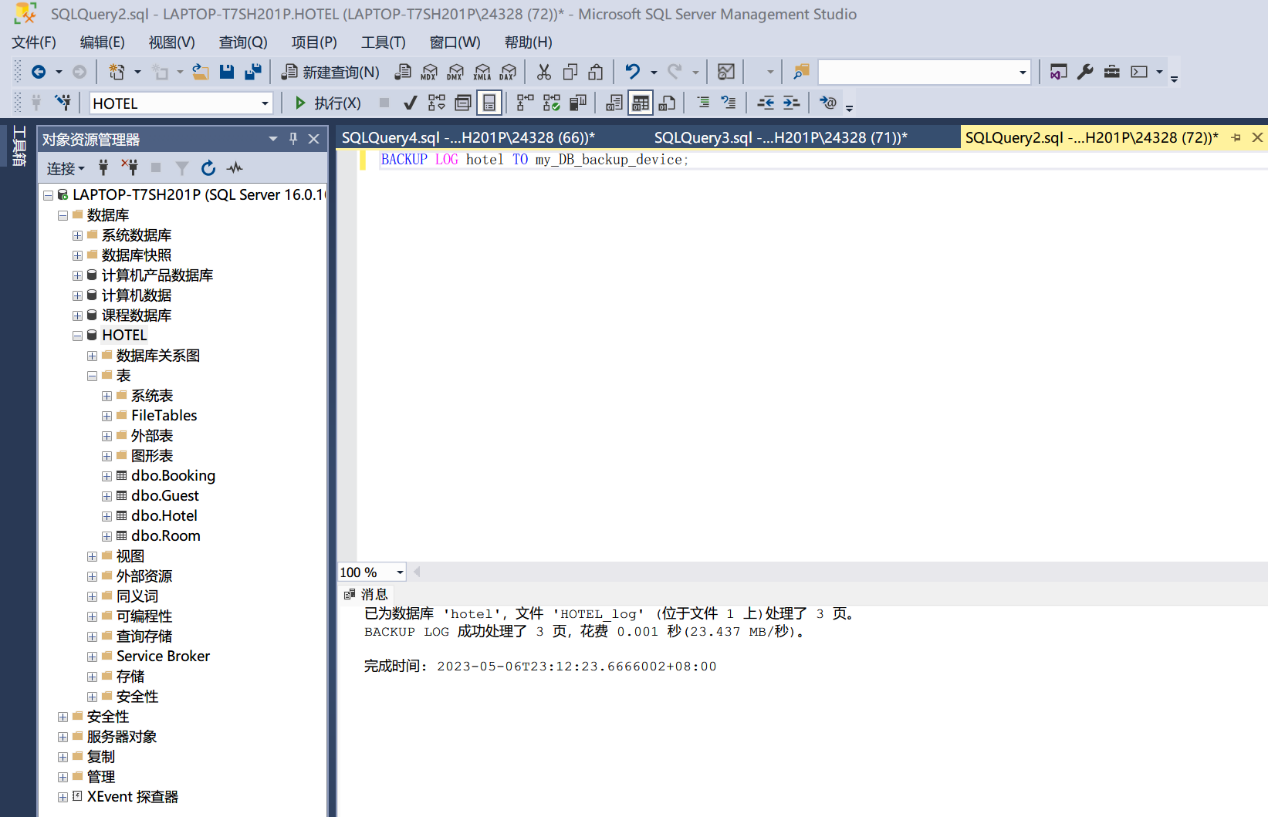


备份到D/DATABASE中

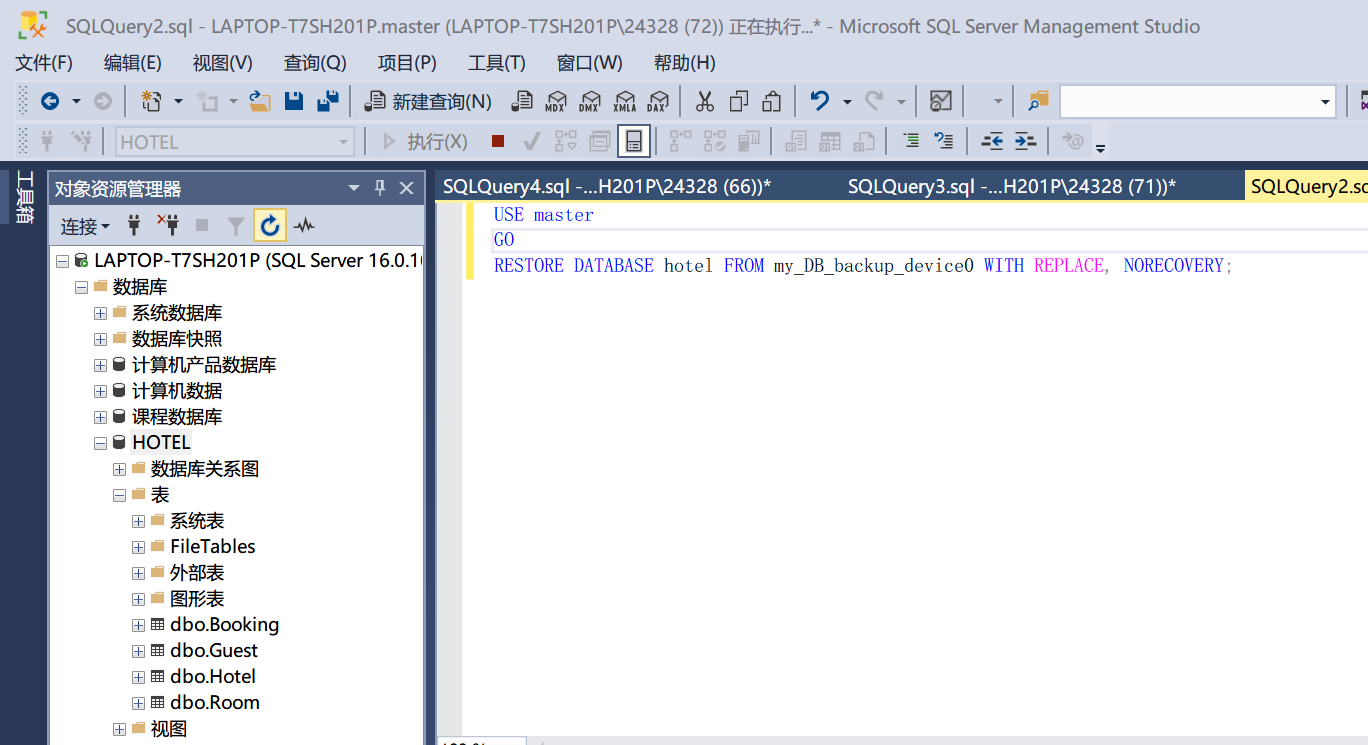
1. 向实验数据库中某个表插入若干条记录。

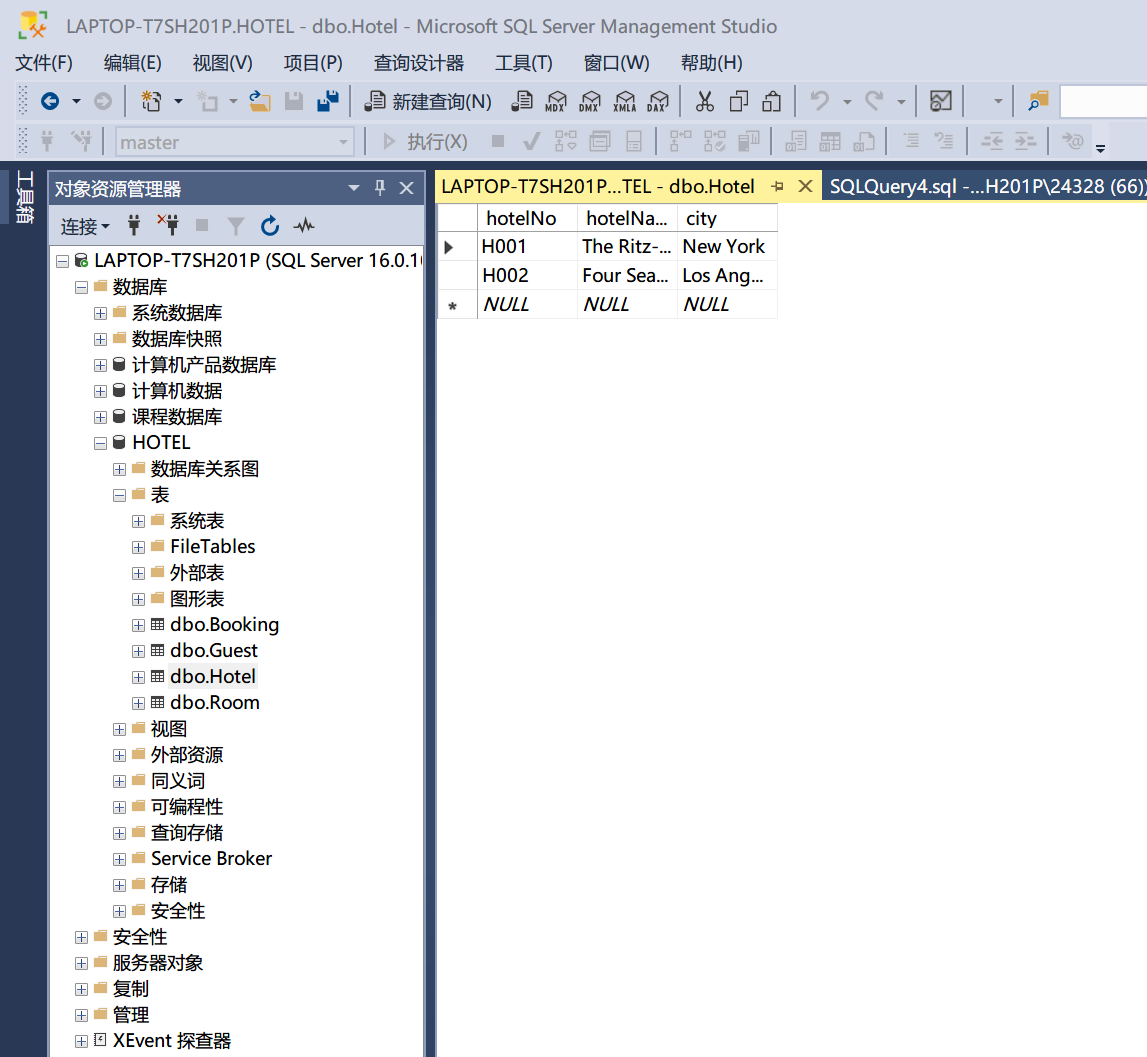


1. 备份数据库事务日志到备份设备，并查看日志格式。



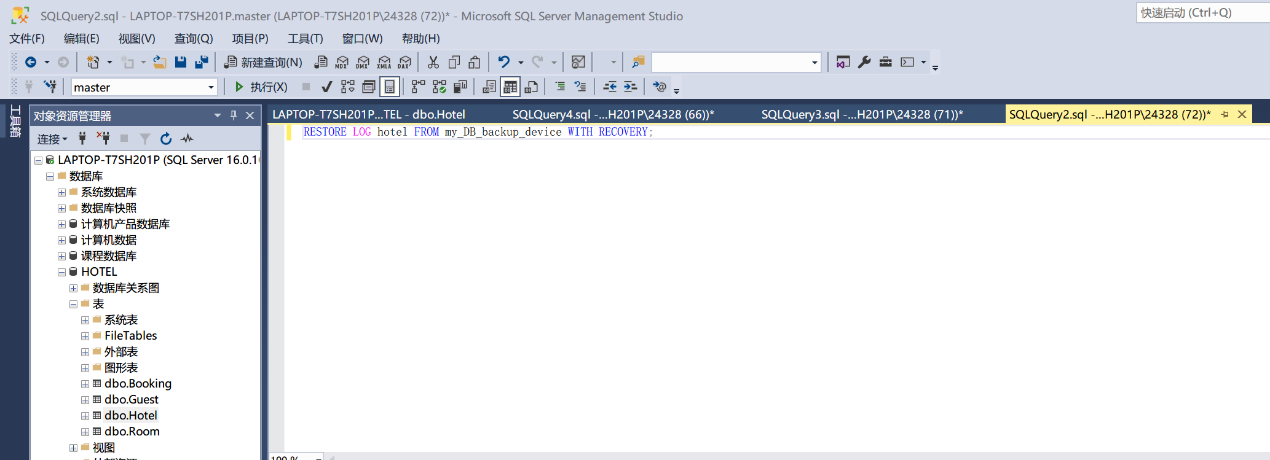
1. 利用第2步所得的完整备份，恢复到插入记录前的状态。

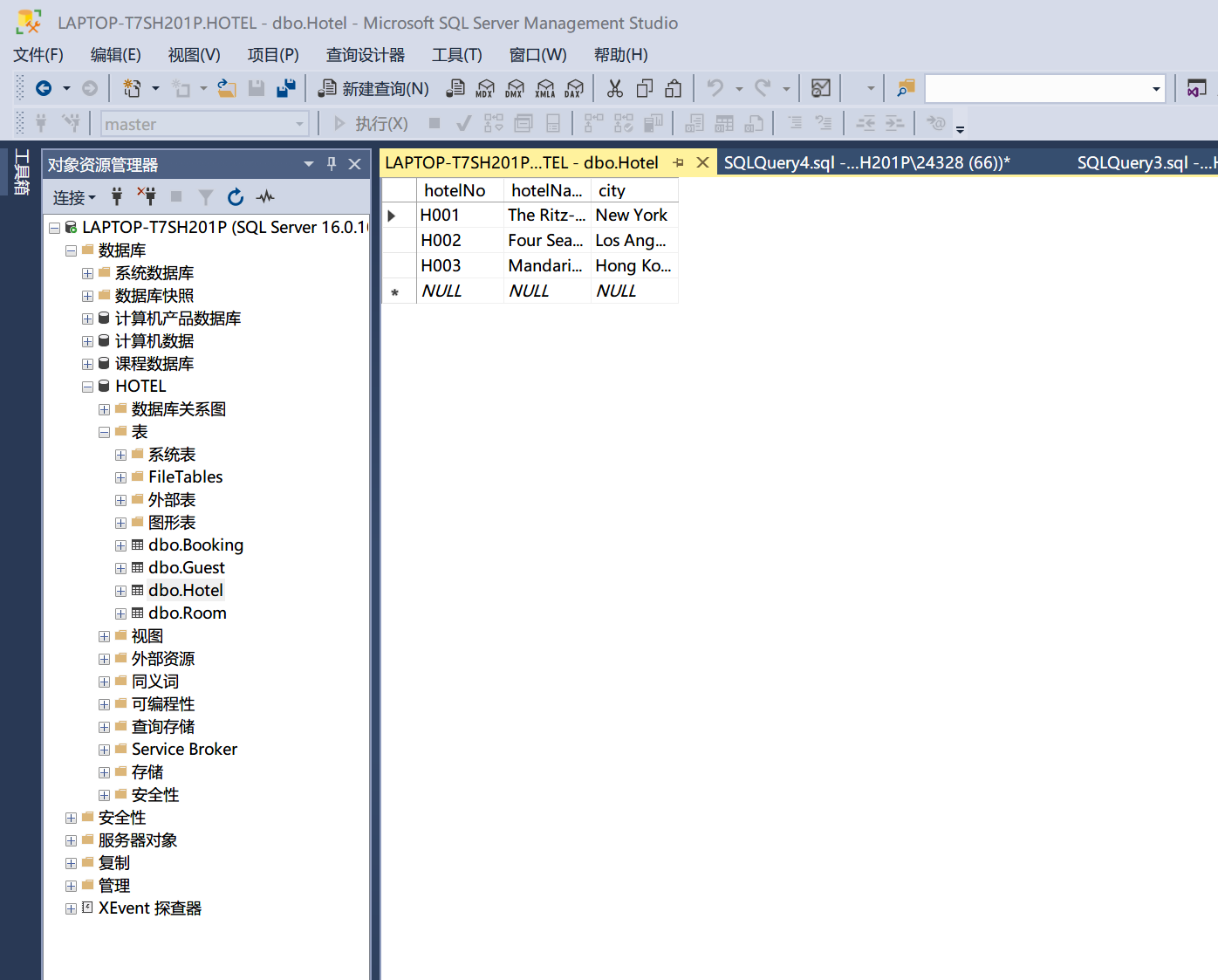




恢复到了插入前的表

(6)利用第4步所得的事务日志，恢复到插入记录后的状态。





恢复到了插入后的表

### 4.触发器

自行为实验数据库的某个表设计一个触发器，实现某种完整性控制，并验证效果

代码：

CREATE TRIGGER CheckHotelNoFormat

ON Hotel

AFTER INSERT

AS

BEGIN

DECLARE @hotelNo VARCHAR(20);

SELECT @hotelNo = hotelNo FROM inserted;

IF (LEN(@hotelNo) <> 9 OR LEFT(@hotelNo, 4) <> 'HTL-' OR

ISNUMERIC(SUBSTRING(@hotelNo, 5, LEN(@hotelNo) - 4)) = 0)

BEGIN

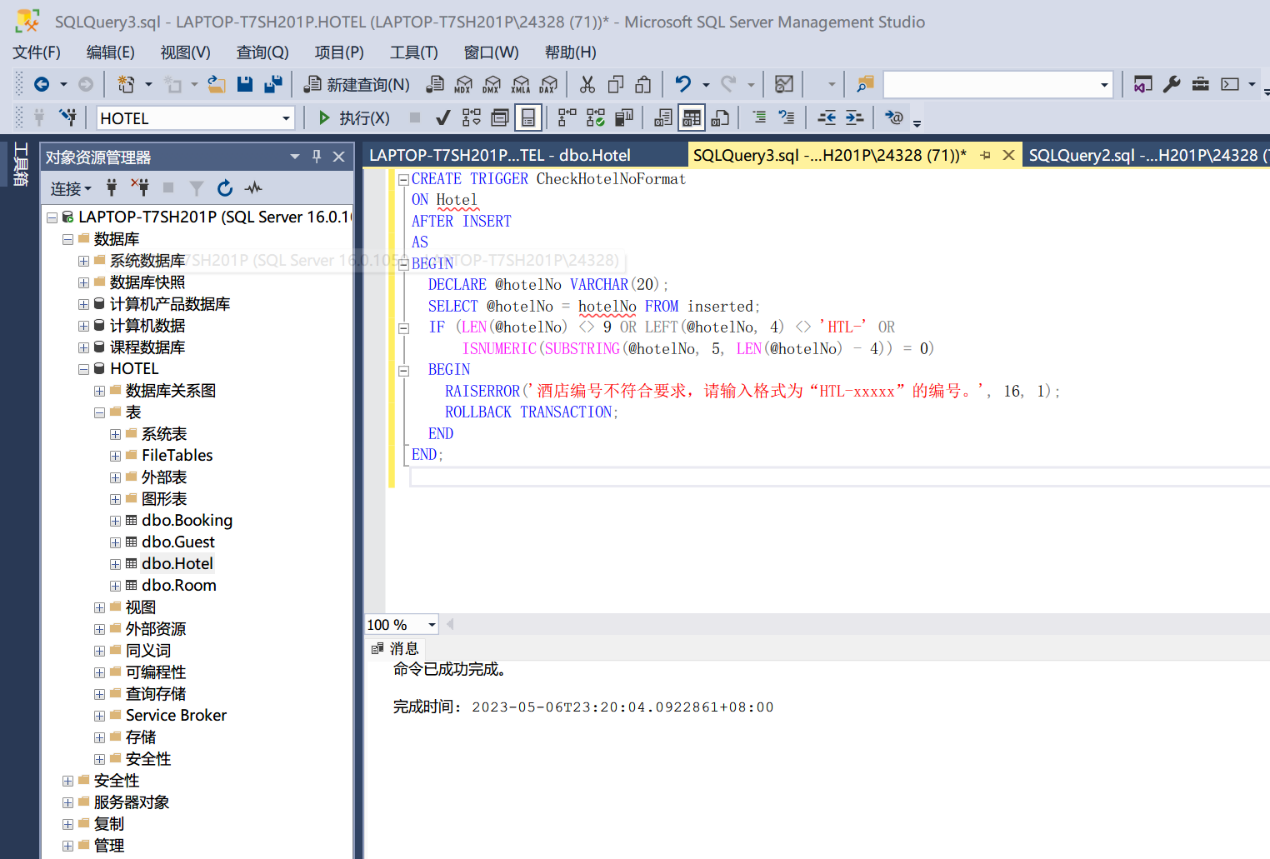
RAISERROR('酒店编号不符合要求，请输入格式为“HTL-xxxxx”的编号。', 16, 1);

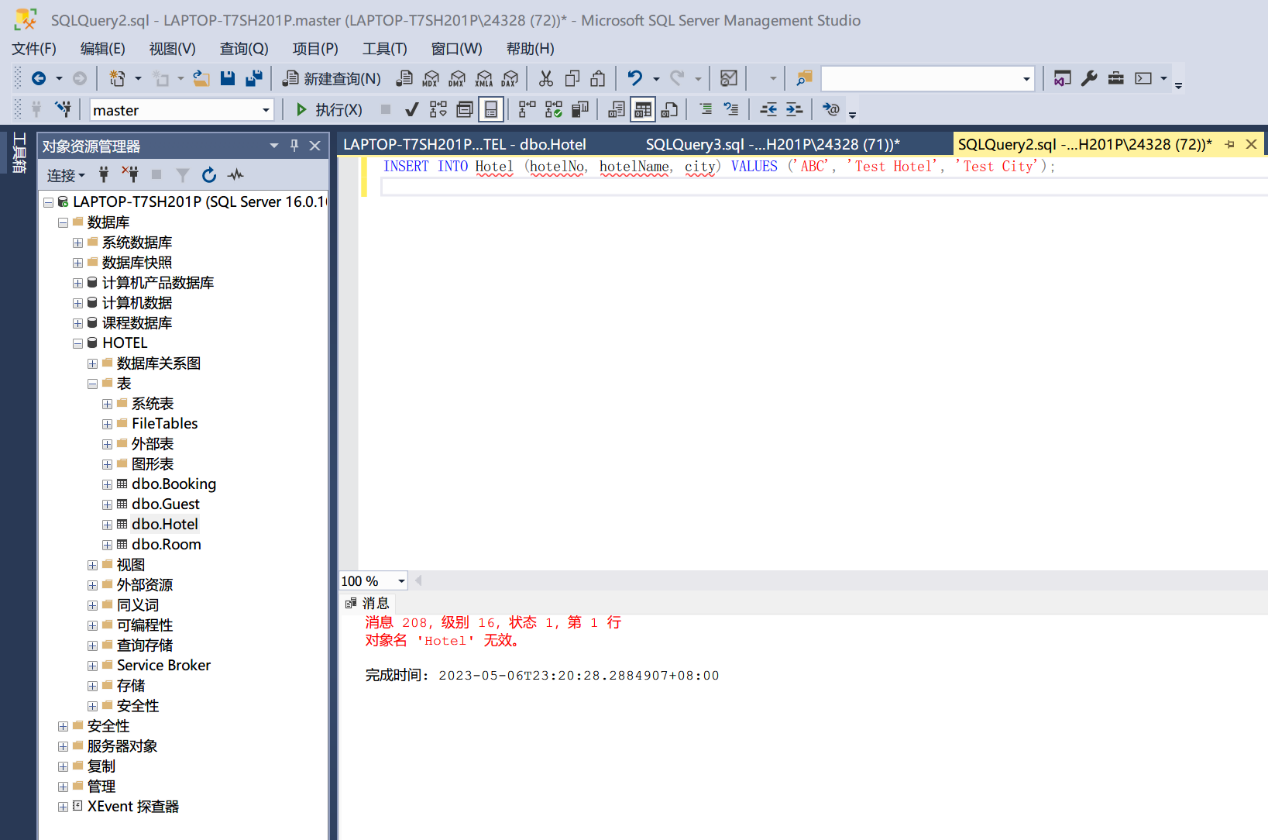
ROLLBACK TRANSACTION;

END

END;

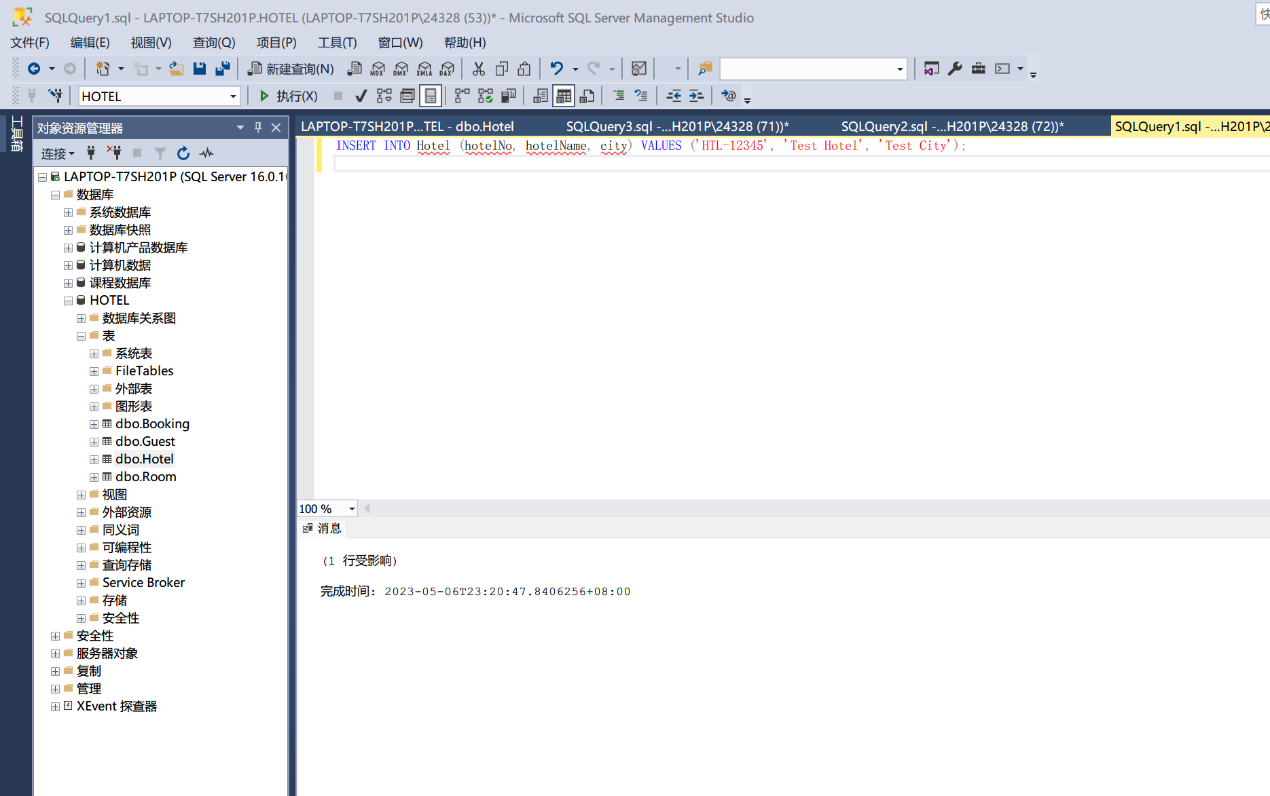
要输入对应格式为“HTL-xxxxx”的编号的hotelNo





INSERT INTO Hotel (hotelNo, hotelName, city) VALUES ('ABC', 'Test Hotel', 'Test City');会错误消息并回滚事务，但是由于sqlserver的原因没有触发。

而INSERT INTO Hotel (hotelNo, hotelName, city) VALUES ('HTL-12345', 'Test Hotel', 'Test City');语句正常。



## 总结

在实验中遇到了不少困难，从什么都不会，到慢慢掌握SQL SERVER，在这其中收获了很多知识，也学会了很多，为我以后的数据库学习打下了坚固的基础。