实验报告 4：SQL 高级编程

课程名称：数据库系统实践	**年级**：2018 级	**上机实践成绩**：
指导教师：赵慧	**姓名**：陈俊潼	
上机实践名称：SQL 实践	**学号**：10185101210	**上机实践日期**：2020.4.30
上机实践编号：4	**组号**：第 12 小组	**上机实践时间**：10:00-11:30

一、目的

- 熟练使用 Transact-SQL 语言创建、删除、更改数据库对象和查询、插入、删除、 更改数据
- 掌握触发器/存储过程的概念和使用方法,使用 Transact-SQL 编写触发器/存储过程

二、内容与设计思想

- 使用 Transact sql 进行简单的编程
- 根据问题需求，编写简单的触发器/存储过程，并调用执行

三、使用环境

macOS 10.15.3
Microsoft SQL Server 2020
Microsoft SQL Server Management Studio 2020

四、实验过程

该部分具有大量代码，为了便于排版和方便阅读，我使用了 Markdown 编写。
见后续页面。

五、总结

经过本次实验，我掌握了 Transact-SQL 的基本使用和编程思想，并通过手动建立数据库和按需求编写脚本了解了触发器和存储过程的概念。

六、附录

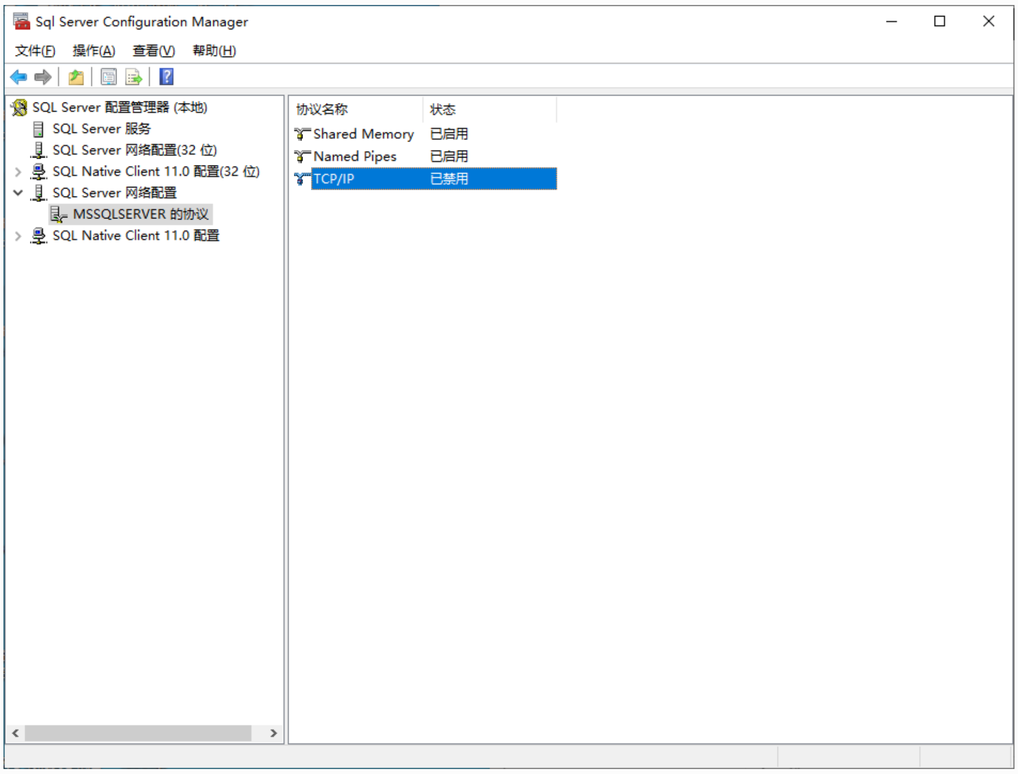
Microsoft SQL 数据导入脚本*（见后续页面）*。

Database LAB 3

> 10185101210 陈俊潼

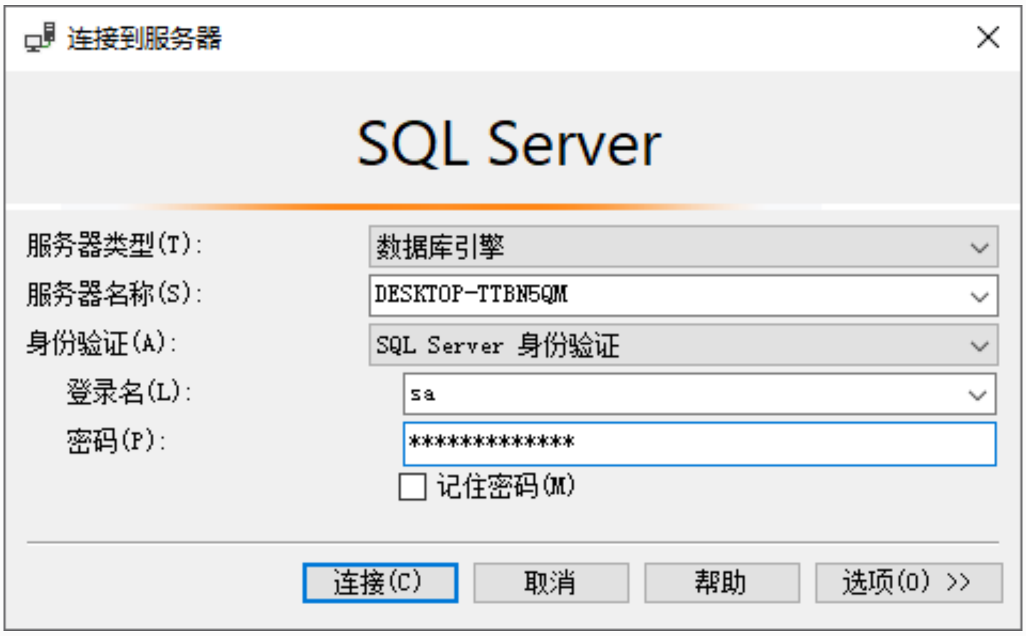
准备工作

在 Microsoft SQL Server 下载客户端并配置安装。安装后配置 SQL Server 的 TCP/IP 协议和 Named Pipes 协议为开启状态，以允许远程连接：


Sql Server Configuration Manager
文件(F) 操作(A) 查看(V) 帮助(H)
SQL Server 配置管理器 (本地)
SQL Server 服务
SQL Server 网络配置(32 位)
SQL Native Client 11.0 配置(32 位)
SQL Server 网络配置
MSSQLSERVER 的协议
SQL Native Client 11.0 配置

协议名称	状态
Shared Memory	已启用
Named Pipes	已启用
TCP/IP	已禁用

接着安装 Microsoft SQL Server Management Studio，启动软件并连接数据库，为 sa 用户指定登录密码。



服务器类型(T)：	数据库引擎
服务器名称(S)：	DESKTOP-TTBN5QM
身份验证(A)：	SQL Server 身份验证
登录名(L)：	sa
密码(P)：	*************
☐ 记住密码(M)	

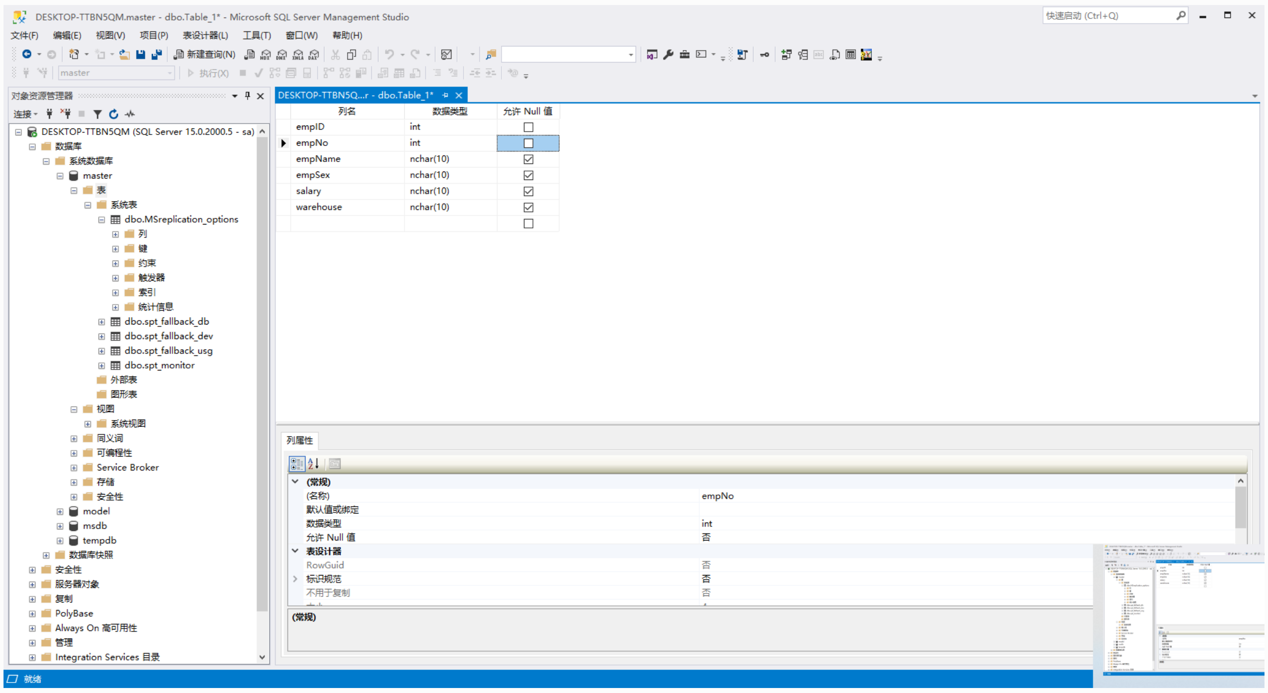
Exercise 1

编写触发器如下:

```sql
CREATE trigger sectionTrigger FOR section ON INSERT
AS
BEGIN
    DECLARE @tid int;
    SELECT @tid = time_slot_id FROM inserted;
    IF NOT EXISTS (SELECT * FROM time_slot WHERE time_slot_id)
    BEGIN
        print 'Referential intergrity not satisfied.'
        ROLLBACK;
    END
END
```

Exercise 2

根据要求，创建数据库employer和warehouse数据库表:



DESKTOP-TTBN5QM.master - dbo.Table_1* - Microsoft SQL Server Management Studio

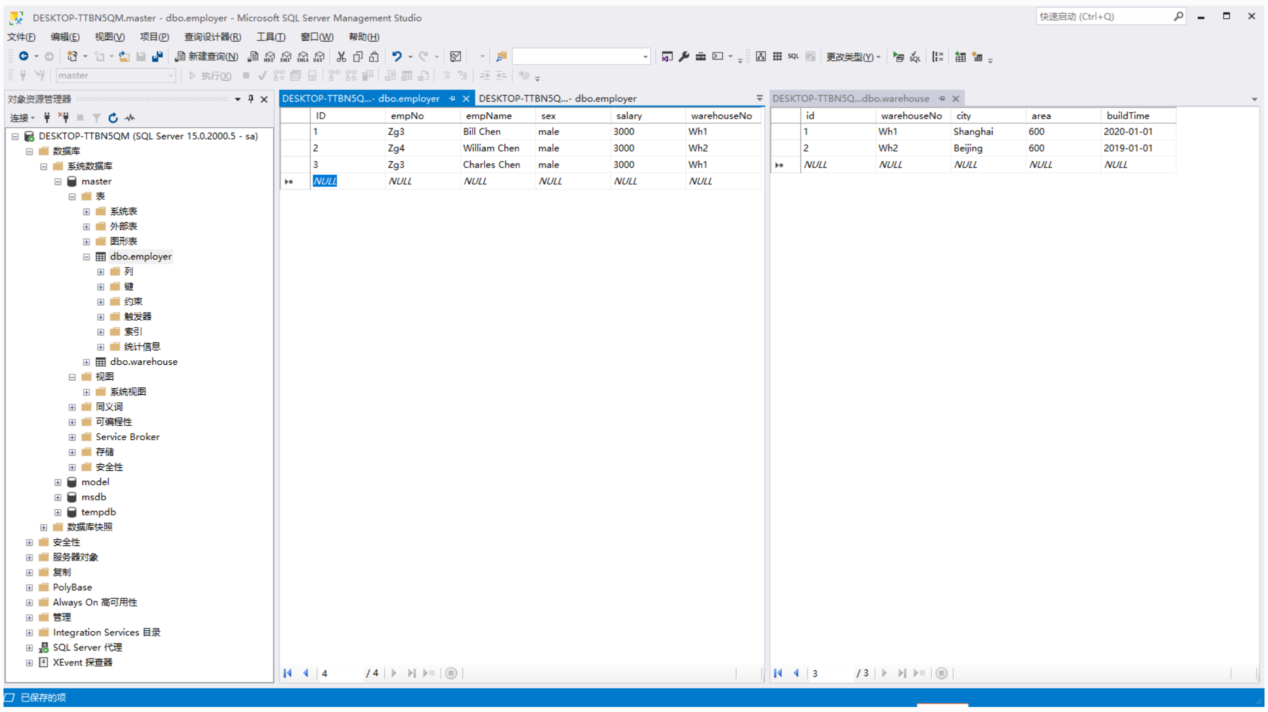
DESKTOP-TTBN5Q...r - dbo.Table_1*

列名	数据类型	允许 Null 值
empID	int	☐
empNo	int	☐
empName	nchar(10)	✓
empSex	nchar(10)	✓
salary	nchar(10)	✓
warehouse	nchar(10)	✓
		☐

列属性

(常规)	
(名称)	empNo
默认值或绑定	
数据类型	int
允许 Null 值	否
表设计器	
RowGuid	否
标识规范	否
不用于复制	否

并构建如图所示的虚拟数据，建立了三个员工，两个仓库:


DESKTOP-TTBN5QM.master - dbo.employer - Microsoft SQL Server Management Studio

DESKTOP-TTBN5Q...- dbo.employer

	ID	empNo	empName	sex	salary	warehouseNo
	1	Zg3	Bill Chen	male	3000	Wh1
	2	Zg4	William Chen	male	3000	Wh2
	3	Zg3	Charles Chen	male	3000	Wh1
▸*	NULL	NULL	NULL	NULL	NULL	NULL

DESKTOP-TTBN5Q...dbo.warehouse

	id	warehouseNo	city	area	buildTime
	1	Wh1	Shanghai	600	2020-01-01
	2	Wh2	Beijing	600	2019-01-01
▸*	NULL	NULL	NULL	NULL	NULL

根据需求，随机建立一些演示数据。这里使用的数据如下：表 employer:

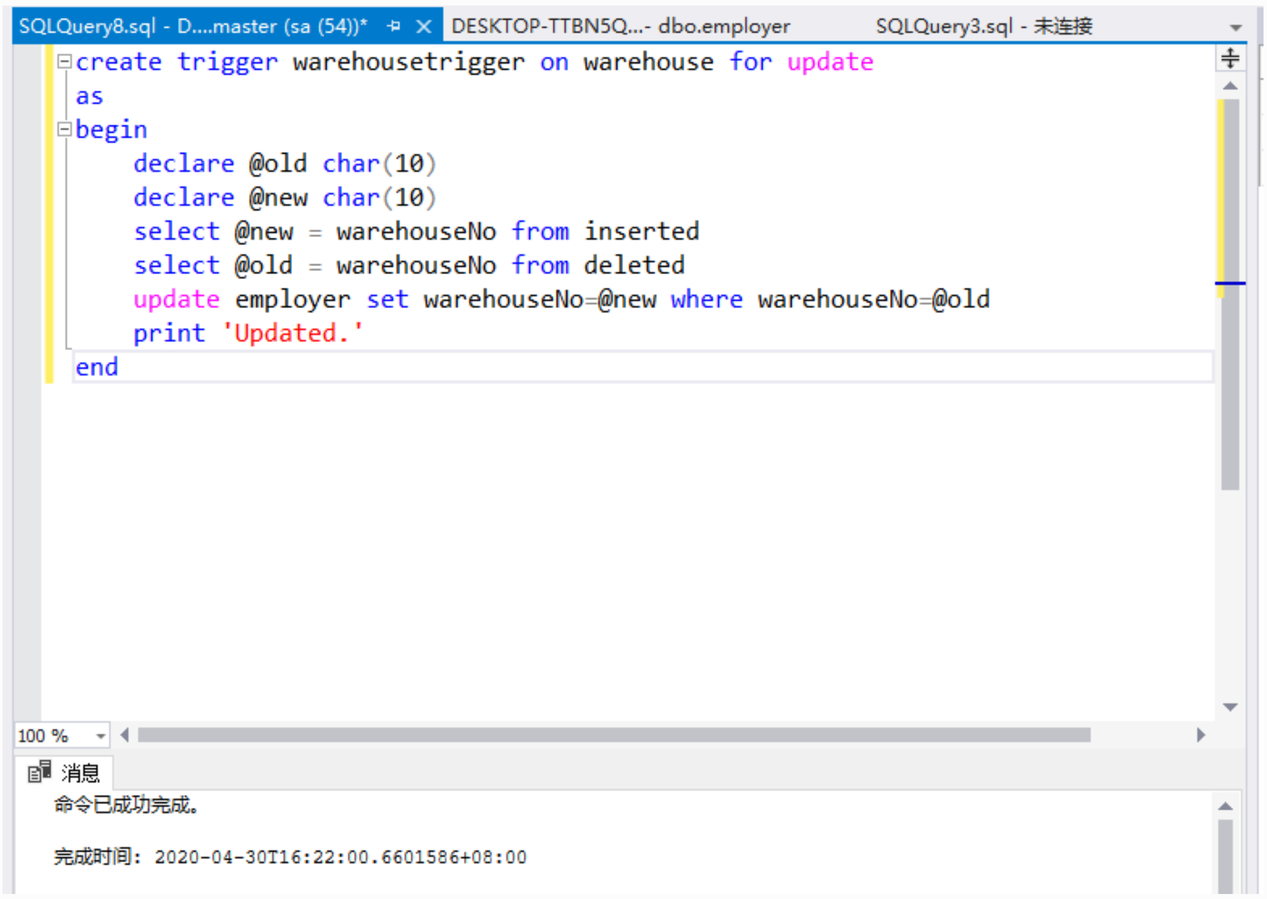
ID	employerNo	name	sex	salary	warehouseNo
1	Zg3	Bill Chen	male	3000	Wh1
2	Zg4	William Chen	male	3000	Wh2
3	Zg3	Charles Chen	male	3000	Wh1
NULL	NULL	NULL	NULL	NULL	NULL

表 warehouse:

ID	warehouseNo	city	area	buildTime
1	Wh1	Shanghai	600	2020-01-01
2	Wh2	Beijing	600	2019-01-01
NULL	NULL	NULL	NULL	NULL

接着为了实现在更新仓库名的时候同步更新雇员表内的仓库名，这里创建 trigger:

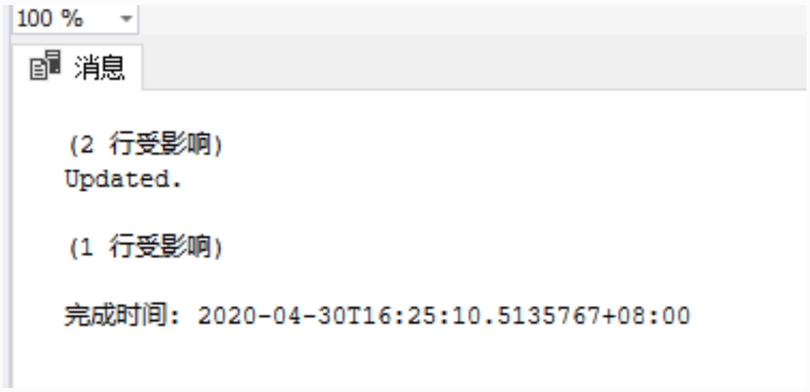
```sql
create trigger warehousetrigger on warehouse for update
as
begin
    declare @old char(10)
    declare @new char(10)
    select @new = warehouseNo from inserted
    select @old = warehouseNo from deleted
    update employer set warehouseNo=@new where warehouseNo=@old
    print 'Updated.'
end
```


SQLQuery8.sql - D....master (sa (54))*
DESKTOP-TTBN5Q...- dbo.employer
SQLQuery3.sql - 未连接
100 %
消息

```sql
create trigger warehousetrigger on warehouse for update
as
begin
    declare @old char(10)
    declare @new char(10)
    select @new = warehouseNo from inserted
    select @old = warehouseNo from deleted
    update employer set warehouseNo=@new where warehouseNo=@old
    print 'Updated.'
end
```

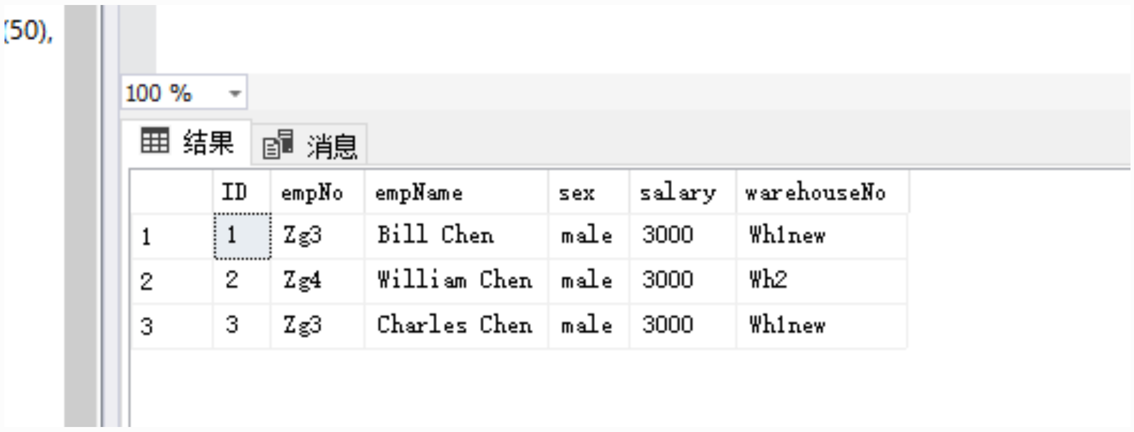
```
命令已成功完成。

完成时间: 2020-04-30T16:22:00.6601586+08:00
```

接下来验证是否成功。执行 `UPDATE warehouse SET warehouseNo='Wh1new' WHERE ID=1;`：



```
(2 行受影响)
Updated.

(1 行受影响)

完成时间: 2020-04-30T16:25:10.5135767+08:00
```

执行成功后，发现 `employer` 表中的信息也随之发生了更新。

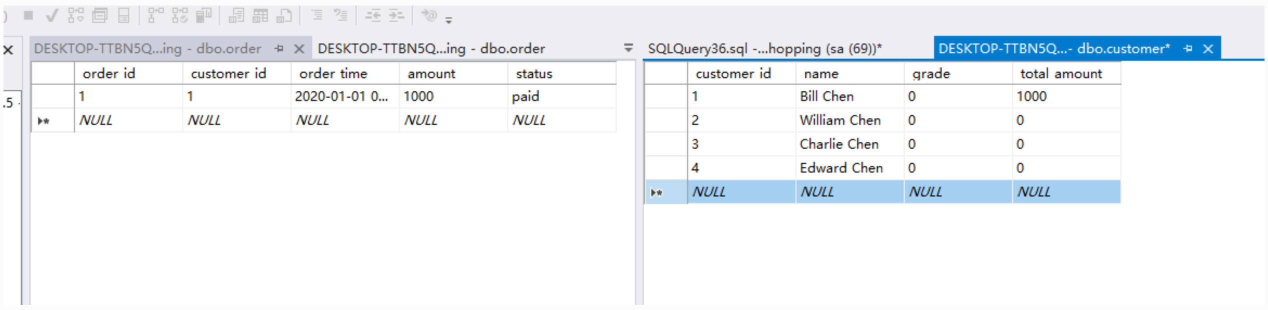


	ID	empNo	empName	sex	salary	warehouseNo
1	1	Zg3	Bill Chen	male	3000	Wh1new
2	2	Zg4	William Chen	male	3000	Wh2
3	3	Zg3	Charles Chen	male	3000	Wh1new

Exercise 3

首先新建两个数据库表。为了简化表示，保留了几个必要字段。对于用户，保留了ID，姓名，等级和总消费额；对于账单，只保留了账单ID，用户ID和，日期，订单价格和签收状态。建立数据关系后，构造简单的示例数据。

这里 grade 作为会员等级，使用 0 表示普通用户，1 表示白金会员，2 表示铂金会员。累计金额满 10000 元 和 20000 元可以升级到白金会员和铂金会员。

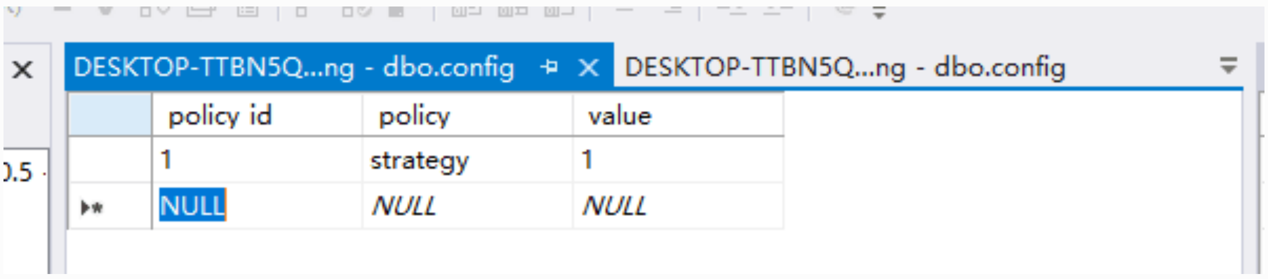

DESKTOP-TTBN5Q...ing - dbo.order

order id	customer id	order time	amount	status
1	1	2020-01-01 0...	1000	paid
NULL	NULL	NULL	NULL	NULL

DESKTOP-TTBN5Q...- dbo.customer*

customer id	name	grade	total amount
1	Bill Chen	0	1000
2	William Chen	0	0
3	Charlie Chen	0	0
4	Edward Chen	0	0
NULL	NULL	NULL	NULL

为了能够实时切换打折策略，这里新建了一个 `config` 表，用于存储策略：


DESKTOP-TTBN5Q...ng - dbo.config

policy id	policy	value
1	strategy	1
NULL	NULL	NULL

这里只存储了一个 `strategy` 行，0 和 1 对应的不同的打折策略。

接下来按照以下逻辑实现打折策略：

startege 值	订单金额累计时间	临界商品是否打折
0	支付前	是
1	支付后	否
2	签收后	否

接下来首先编写修改策略的控制存储：

```sql
CREATE proc changeStrategy
@strategy int
AS
BEGIN
    UPDATE config SET value = @strategy WHERE policy = 'strategy';
    print 'Stratedy updated.'
END
```

现在使用 `execute changeStrategy x` 即可更改策略。

针对不同的策略，在插入新订单的时候会有不同的升级措施和打折措施。于是编写针对 `order` 的 trigger 如下：

```sql
CREATE TRIGGER orderInsertTrigger ON [order] FOR INSERT
AS
```

```sql
BEGIN
    DECLARE @s int;
    DECLARE @uid int;
    DECLARE @sum float;
    DECLARE @thisamount float;
    SELECT @s = value FROM config WHERE policy = 'strategy';
    SELECt @uid = customer_id FROM inserted
    SELECT @thisamount = amount FROM inserted
    IF @s = 0
    BEGIN
        --- Count before payment
        SELECT @sum=sum(amount) FROM [order] WHERE customer_id = @uid;
        IF @sum >= 20000
        BEGIN
            UPDATE customer SET grade=2 WHERE customer_id = @uid;
            print 'Customer grade set to 2.'
        END
        ELSE IF @sum >= 10000
        BEGIN
            UPDATE customer SET grade=1 WHERE customer_id = @uid;
            print 'Customer grade set to 1.'
        END
    END
    ELSE IF @s = 1
    BEGIN
        --- Count after payment
        SELECT @sum=sum(amount) FROM [order] WHERE customer_id = @uid AND
        ([status] = 'paid' OR [status] = 'delivered');
        IF @sum >= 20000
        BEGIN
            UPDATE customer SET grade=2 WHERE customer_id = @uid;
        END
        ELSE IF @sum >= 10000
        BEGIN
            UPDATE customer SET grade=1 WHERE customer_id = @uid;
        END
    END
    ELSE IF @s = 2
    BEGIN
        --- Count after delivered
        SELECT @sum=sum(amount) FROM [order] WHERE customer_id = @uid AND
[status] = 'paid';
        IF @sum >= 20000
        BEGIN
            UPDATE customer SET grade=2 WHERE customer_id = @uid;
            print 'Customer grade set to 2.'
        END
        ELSE IF @sum >= 10000
        BEGIN
            UPDATE customer SET grade=1 WHERE customer_id = @uid;
            print 'Customer grade set to 1.'
        END
    END
    UPDATE customer SET total_amount = @sum WHERE customer_id = @uid;
END
```

以上是针对插入的 trigger。考虑到第三种策略需要在用户签收后再计算订单总额，再针对第三种策略新建一个针对更新的 trigger：

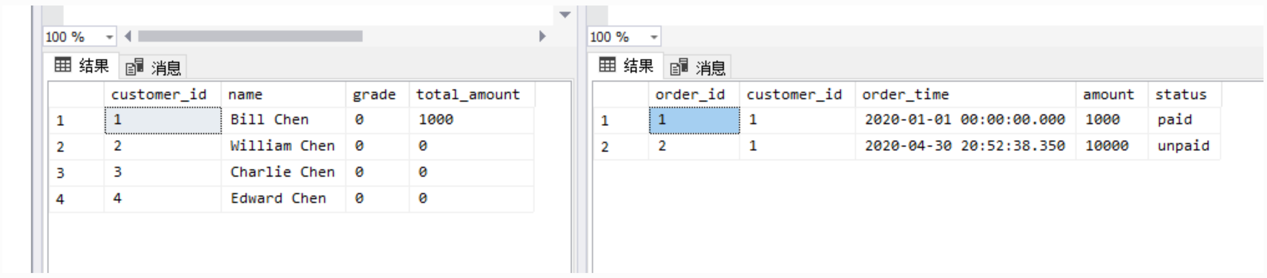
```
CREATE TRIGGER orderUpdateTrigger ON [order] FOR UPDATE
AS
BEGIN
    DECLARE @s int;
    DECLARE @uid int;
    DECLARE @sum float;
    DECLARE @thisamount float;
    DECLARE @newstatus varchar(50);
    SELECT @s = value FROM config WHERE policy = 'strategy';
    SELECt @uid = customer_id FROM inserted
    SELECT @thisamount = amount FROM inserted
    SELECT @newstatus = status FROM inserted
    print 'Update trigger running...'
    IF @s = 2 AND @newstatus = 'delivered'
    BEGIN
        --- Update after delivered
        SELECT @sum=sum(amount) FROM [order] WHERE customer_id = @uid AND status
= 'delivered';
        print 'Order delivered.'
        IF @sum >= 20000
        BEGIN
            UPDATE customer SET grade=2 WHERE customer_id = @uid;
            print 'Customer grade set to 2.'
        END
        ELSE IF @sum >= 10000
        BEGIN
            UPDATE customer SET grade=1 WHERE customer_id = @uid;
            print 'Customer grade set to 1.'
        END
        UPDATE customer SET total_amount = @sum WHERE customer_id = @uid;
    END
END
```

完成后，我们将策略设置成 2，即在订单签收后更新总额。

执行以下 SQL 语句：

```
EXECUTE changeStrategy 2;

INSERT INTO [order] (order_id, customer_id, order_time, amount, status)  VALUES
(2, 1, CURRENT_TIMESTAMP, 10000, 'unpaid');
```

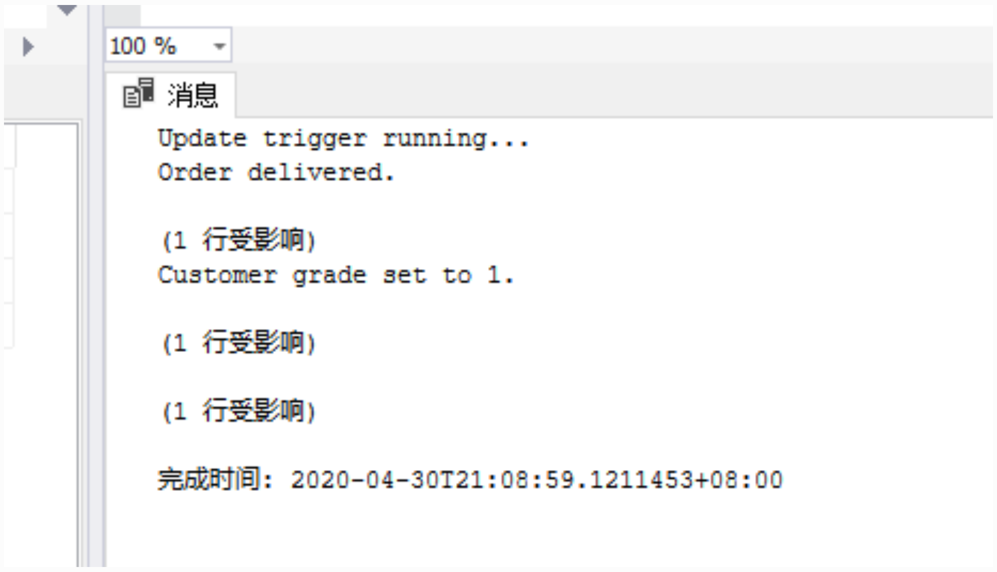
此时发现用户表中的账单总数没有更新，等级仍然是 0。



	customer_id	name	grade	total_amount
1	1	Bill Chen	0	1000
2	2	William Chen	0	0
3	3	Charlie Chen	0	0
4	4	Edward Chen	0	0

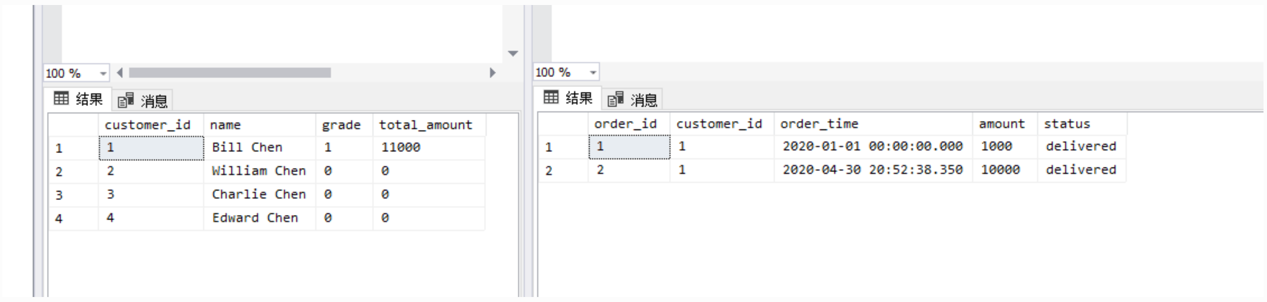
	order_id	customer_id	order_time	amount	status
1	1	1	2020-01-01 00:00:00.000	1000	paid
2	2	1	2020-04-30 20:52:38.350	10000	unpaid

接着执行`UPDATE [order] SET status='delivered' WHERE order_id = 2;`，模拟订单签收操作。可以看到，已经触发了触发器。



```
Update trigger running...
Order delivered.

(1 行受影响)
Customer grade set to 1.

(1 行受影响)

(1 行受影响)

完成时间: 2020-04-30T21:08:59.1211453+08:00
```

再次查询 `order` 和 `customer` 表的信息，可以看见已经得到了更新：



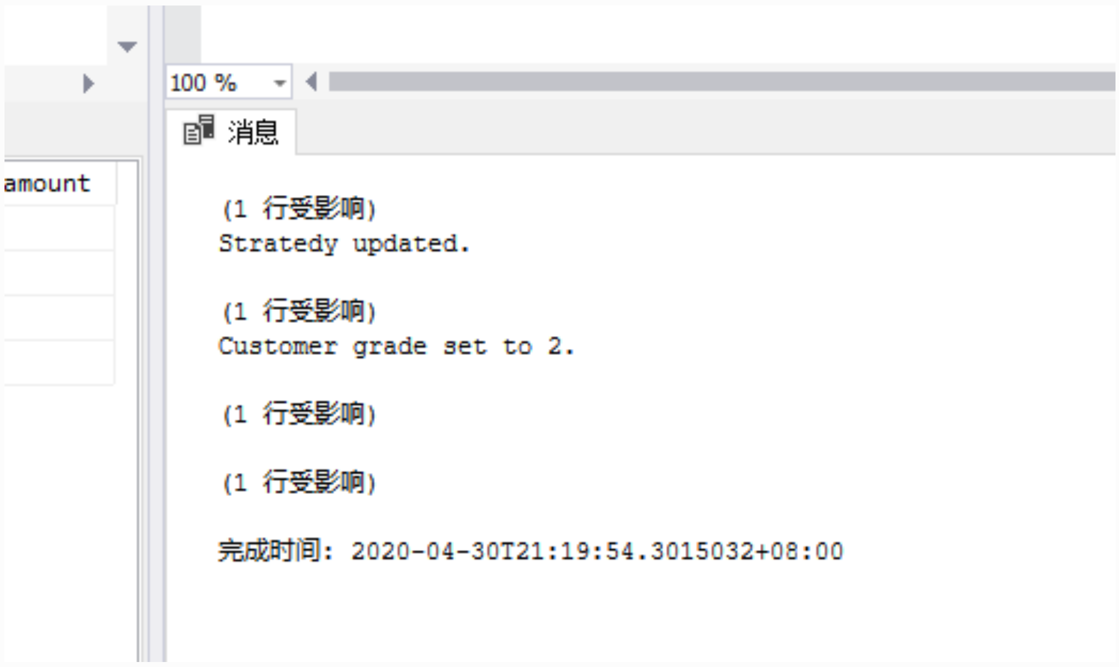
	customer_id	name	grade	total_amount
1	1	Bill Chen	1	11000
2	2	William Chen	0	0
3	3	Charlie Chen	0	0
4	4	Edward Chen	0	0

	order_id	customer_id	order_time	amount	status
1	1	1	2020-01-01 00:00:00.000	1000	delivered
2	2	1	2020-04-30 20:52:38.350	10000	delivered

再次把策略设置为 0，即只要提交订单就计算总额。运行以下命令：

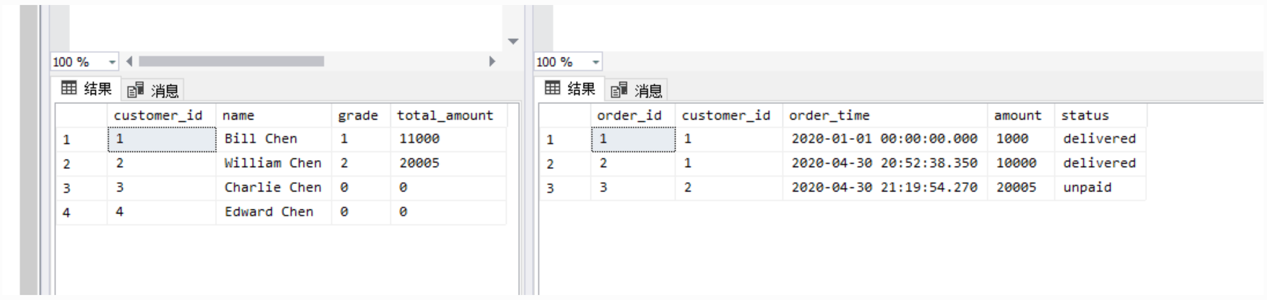
```
EXECUTE changeStrategy 0;

INSERT INTO [order] (order_id, customer_id, order_time, amount, status)  VALUES
(3, 2, CURRENT_TIMESTAMP, 20005, 'unpaid');
```

可以看到订单还未签收，用户等级已经更新到铂金会员：



```
(1 行受影响)
Stratedy updated.

(1 行受影响)
Customer grade set to 2.

(1 行受影响)

(1 行受影响)

完成时间: 2020-04-30T21:19:54.3015032+08:00
```

此时两个表的查询数据结果如下:

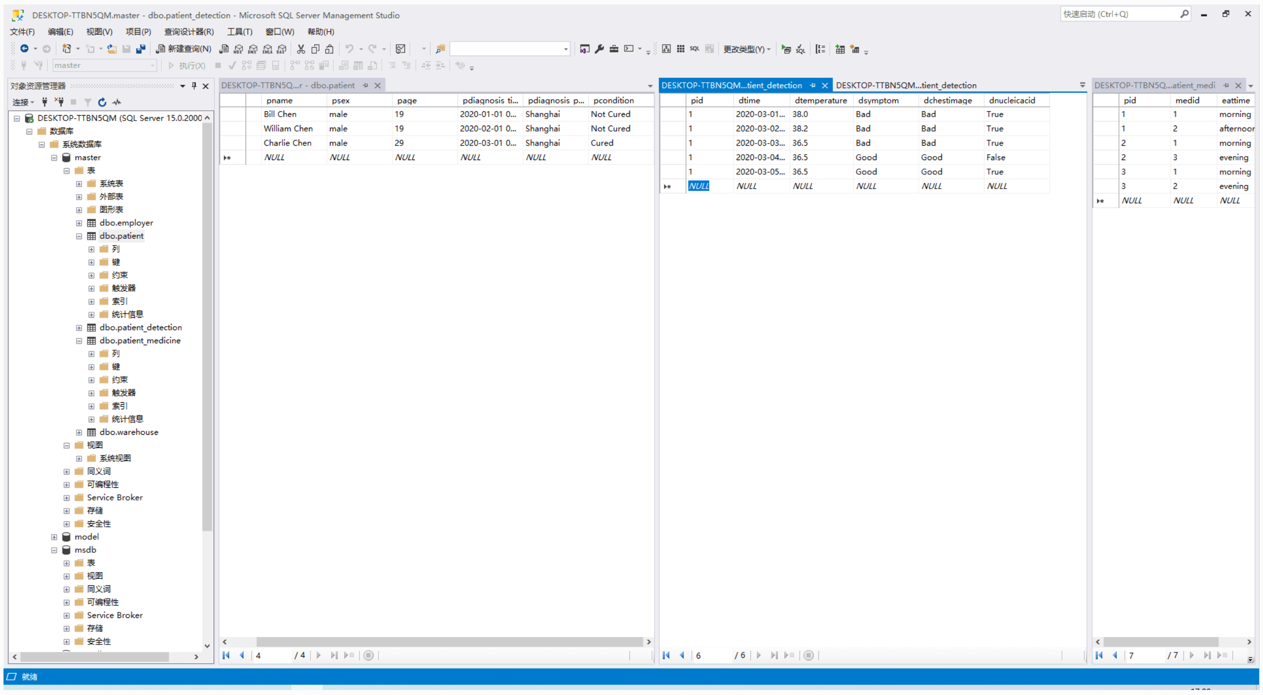


	customer_id	name	grade	total_amount
1	1	Bill Chen	1	11000
2	2	William Chen	2	20005
3	3	Charlie Chen	0	0
4	4	Edward Chen	0	0

	order_id	customer_id	order_time	amount	status
1	1	1	2020-01-01 00:00:00.000	1000	delivered
2	2	1	2020-04-30 20:52:38.350	10000	delivered
3	3	2	2020-04-30 21:19:54.270	20005	unpaid

Exercise 4

首先建立数据库表，并虚拟数据:



pname	psex	page	pdiagnosis ti...	pdiagnosis p...	pcondition
Bill Chen	male	19	2020-01-01 0...	Shanghai	Not Cured
William Chen	male	19	2020-02-01 0...	Shanghai	Not Cured
Charlie Chen	male	29	2020-03-01 0...	Shanghai	Cured
NULL	NULL	NULL	NULL	NULL	NULL

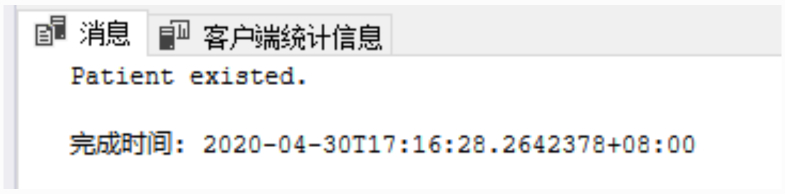
pid	dtime	dtemperature	dsymptom	dchestimage	dnucleicacid
1	2020-03-01...	38.0	Bad	Bad	True
1	2020-03-02...	38.2	Bad	Bad	True
1	2020-03-03...	36.5	Bad	Bad	True
1	2020-03-04...	36.5	Good	Good	False
1	2020-03-05...	36.5	Good	Good	True
NULL	NULL	NULL	NULL	NULL	NULL

pid	medid	eattime
1	1	morning
1	2	afternoon
2	1	morning
2	3	evening
3	1	morning
3	2	evening
NULL	NULL	NULL

仅供测试使用，这里的身份证号用了一位数字代替，对于症状的描述也简化成了Good 和 Bad 两种情况。认为体温良好的范围为 36.0 - 37.5 摄氏度，同时新增加的病例默认会在早晨服用 ID 为 1 的药物。于是编写存储过程如下：

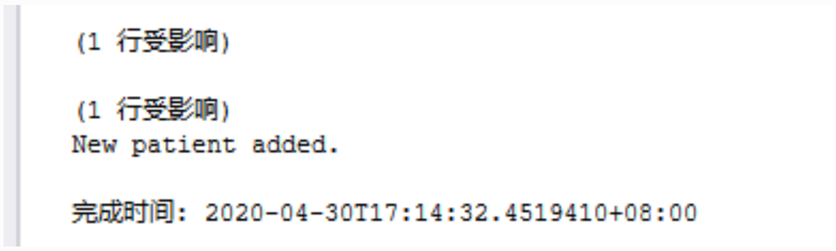
```sql
CREATE proc [dbo].[newPatient]
@pid int
AS
BEGIN
    IF NOT EXISTS (SELECT * FROM patient WHERE pid = @pid)
        BEGIN
            INSERT INTO patient(pid, pname, psex, page, pdiagnosis_time, pdiagnosis_place, pcondition) VALUES (@pid, 'unknown', 'unknown', 'unknown', CURRENT_TIMESTAMP, 'Shanghai', 'Not Cured');
            INSERT INTO patient_medicine(pid, medid, eattime) VALUES (@pid, 1, 'morning')
            print 'New patient added.'
        END
    ELSE
        print 'Patient existed.'
END
```

执行 `execute newPatient 1` 会得到：



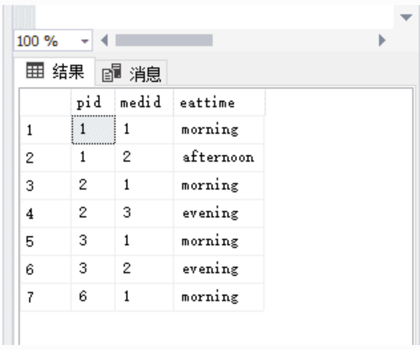
```
Patient existed.

完成时间: 2020-04-30T17:16:28.2642378+08:00
```

此时再执行 `execute newPatient 6` 会得到：

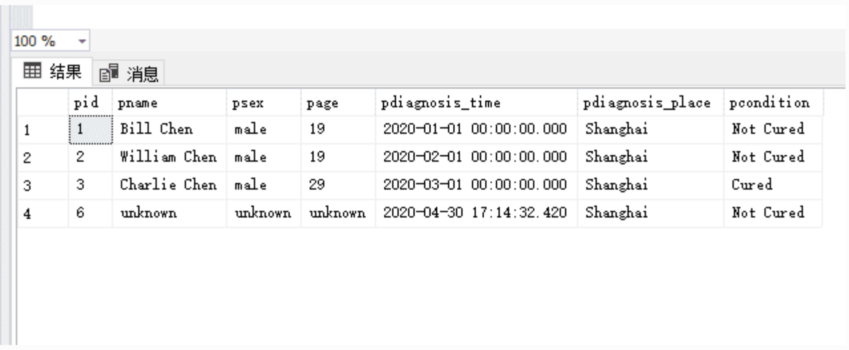


```
(1 行受影响)

(1 行受影响)
New patient added.

完成时间: 2020-04-30T17:14:32.4519410+08:00
```

同时 `patient` 表和 `patient_medicine` 表中的数据都会得到更新：



	pid	medid	eattime
1	1	1	morning
2	1	2	afternoon
3	2	1	morning
4	2	3	evening
5	3	1	morning
6	3	2	evening
7	6	1	morning



	pid	pname	psex	page	pdiagnosis_time	pdiagnosis_place	pcondition
1	1	Bill Chen	male	19	2020-01-01 00:00:00.000	Shanghai	Not Cured
2	2	William Chen	male	19	2020-02-01 00:00:00.000	Shanghai	Not Cured
3	3	Charlie Chen	male	29	2020-03-01 00:00:00.000	Shanghai	Cured
4	6	unknown	unknown	unknown	2020-04-30 17:14:32.420	Shanghai	Not Cured

接着编写触发器如下：

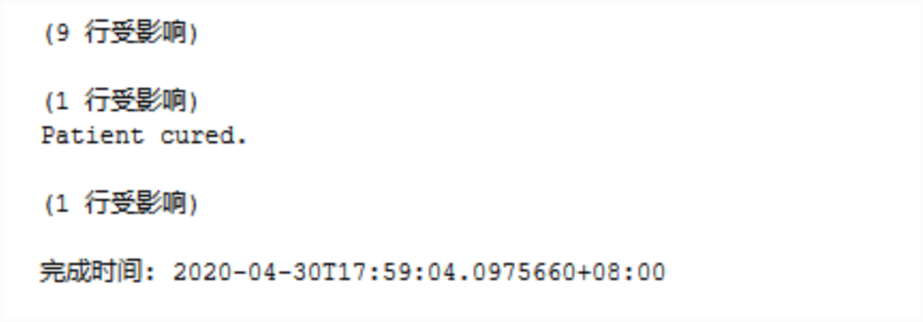
```sql
CREATE trigger [dbo].[diagnosisTrigger] ON [dbo].[patient_detection] FOR INSERT
AS
BEGIN
```

```sql
DECLARE @count int;
DECLARE @pid int;
SELECT @pid = pid FROM inserted;
DROP TABLE #normalDays;
SELECT * INTO #normalDays FROM patient_detection WHERE pid=1 AND dtemperature>=36.0 AND dtemperature<=37.5;
IF @pid IN (SELECT a.pid FROM #normalDays a, #normalDays b, #normalDays c WHERE DATEDIFF(DAY, b.dtime, a.dtime)=1 AND DATEDIFF(DAY, c.dtime, b.dtime)=1
    AND c.dsymptom='Good' AND c.dchestimage='Good' AND b.dnucleicacid='False' AND c.dnucleicacid='False')
    BEGIN
        UPDATE patient SET pcondition='Cured' WHERE pid=@pid;
        print 'Patient cured.'
    END
DROP TABLE #normalDays;
END
```

此时，执行：

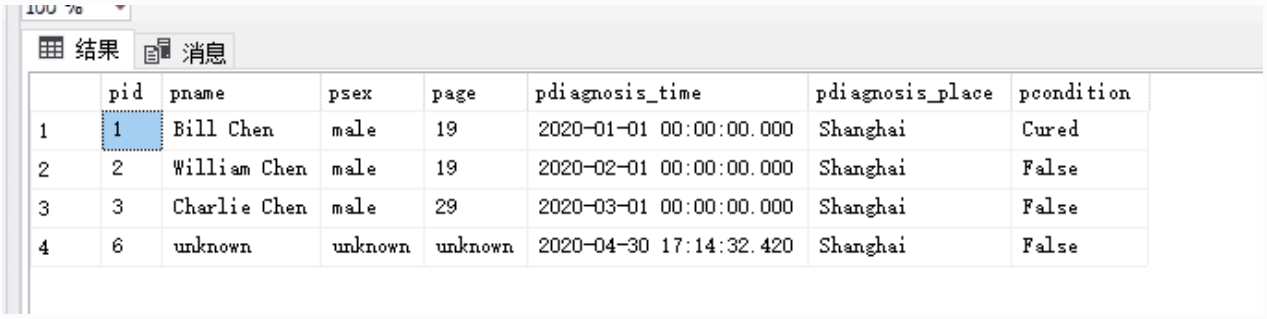
```sql
INSERT into patient_detection(pid, dtime, dtemperature, dsymptom, dchestimage, dnucleicacid)
VALUES (1, '2020-03-11', 36.5, 'Good', 'Good', 'False');
```

可以看到来自 trigger 的提示：



```
(9 行受影响)

(1 行受影响)
Patient cured.

(1 行受影响)

完成时间: 2020-04-30T17:59:04.0975660+08:00
```

病患情况也已经更新成 Cured。



	pid	pname	psex	page	pdiagnosis_time	pdiagnosis_place	pcondition
1	1	Bill Chen	male	19	2020-01-01 00:00:00.000	Shanghai	Cured
2	2	William Chen	male	19	2020-02-01 00:00:00.000	Shanghai	False
3	3	Charlie Chen	male	29	2020-03-01 00:00:00.000	Shanghai	False
4	6	unknown	unknown	unknown	2020-04-30 17:14:32.420	Shanghai	False

---

附录

练习 1、2、4使用的表导出的脚本（位于 master 数据库内）：

```sql
USE [master]
GO
```

```sql
/****** Object:  User [##MS_PolicyEventProcessingLogin##]    Script Date: 2020/4/30 21:24:40 ******/
CREATE USER [##MS_PolicyEventProcessingLogin##] FOR LOGIN [##MS_PolicyEventProcessingLogin##] WITH DEFAULT_SCHEMA=[dbo]
GO
/****** Object:  User [##MS_AgentSigningCertificate##]    Script Date: 2020/4/30 21:24:40 ******/
CREATE USER [##MS_AgentSigningCertificate##] FOR LOGIN [##MS_AgentSigningCertificate##]
GO
/****** Object:  Table [dbo].[employer]    Script Date: 2020/4/30 21:24:40 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[employer](
    [ID] [int] NOT NULL,
    [empNo] [varchar](50) NOT NULL,
    [empName] [varchar](50) NULL,
    [sex] [nchar](10) NULL,
    [salary] [nchar](10) NULL,
    [warehouseNo] [nchar](10) NULL
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[normalDays]    Script Date: 2020/4/30 21:24:40 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[normalDays](
    [pid] [nchar](10) NULL,
    [dtime] [datetime] NULL,
    [dtemperature] [nchar](10) NULL,
    [dsymptom] [varchar](50) NULL,
    [dchestimage] [varchar](50) NULL,
    [dnucleicacid] [varchar](50) NULL
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[patient]    Script Date: 2020/4/30 21:24:40 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[patient](
    [pid] [nchar](10) NULL,
    [pname] [varchar](50) NULL,
    [psex] [nchar](10) NULL,
    [page] [nchar](10) NULL,
    [pdiagnosis_time] [datetime] NULL,
    [pdiagnosis_place] [varchar](50) NULL,
    [pcondition] [nchar](10) NULL
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[patient_detection]    Script Date: 2020/4/30 21:24:40 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[patient_detection](
    [pid] [nchar](10) NULL,
    [dtime] [datetime] NULL,
    [dtemperature] [nchar](10) NULL,
    [dsymptom] [varchar](50) NULL,
    [dchestimage] [varchar](50) NULL,
    [dnucleicacid] [varchar](50) NULL
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[patient_medicine]    Script Date: 2020/4/30 21:24:40 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[patient_medicine](
    [pid] [nchar](10) NULL,
    [medid] [nchar](10) NULL,
    [eattime] [varchar](50) NULL
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[warehouse]    Script Date: 2020/4/30 21:24:40 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[warehouse](
    [ID] [int] NULL,
    [warehouseNo] [varchar](50) NULL,
    [city] [varchar](50) NULL,
    [area] [int] NULL,
    [buildTime] [date] NULL
) ON [PRIMARY]
GO
/****** Object:  StoredProcedure [dbo].[newPatient]    Script Date: 2020/4/30 21:24:40 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE proc [dbo].[newPatient]
```

```sql
@pid int
AS
BEGIN
    IF NOT EXISTS (SELECT * FROM patient WHERE pid = @pid)
        BEGIN
            INSERT INTO patient(pid, pname, psex, page, pdiagnosis_time, pdiagnosis_place, pcondition) VALUES (@pid, 'unknown', 'unknown', 'unknown', CURRENT_TIMESTAMP, 'Shanghai', 'Not Cured');
            INSERT INTO patient_medicine(pid, medid, eattime) VALUES (@pid, 1, 'morning')
            print 'New patient added.'
        END
    ELSE
        print 'Patient existed.'
END
GO
```

练习 3 中的数表生成脚本（位于 shopping 数据库内）：

```sql
USE [master]
GO
/****** Object:  Database [shopping]    Script Date: 2020/4/30 21:22:09 ******/
CREATE DATABASE [shopping]
 CONTAINMENT = NONE
 ON  PRIMARY
( NAME = N'shopping', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\shopping.mdf' , SIZE = 8192KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB )
 LOG ON
( NAME = N'shopping_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\shopping_log.ldf' , SIZE = 8192KB , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )
 WITH CATALOG_COLLATION = DATABASE_DEFAULT
GO
ALTER DATABASE [shopping] SET COMPATIBILITY_LEVEL = 150
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [shopping].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO
ALTER DATABASE [shopping] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [shopping] SET ANSI_NULLS OFF
GO
ALTER DATABASE [shopping] SET ANSI_PADDING OFF
GO
ALTER DATABASE [shopping] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [shopping] SET ARITHABORT OFF
```

```sql
GO
ALTER DATABASE [shopping] SET AUTO_CLOSE OFF
GO
ALTER DATABASE [shopping] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [shopping] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [shopping] SET CURSOR_CLOSE_ON_COMMIT OFF
GO
ALTER DATABASE [shopping] SET CURSOR_DEFAULT  GLOBAL
GO
ALTER DATABASE [shopping] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [shopping] SET NUMERIC_ROUNDABORT OFF
GO
ALTER DATABASE [shopping] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [shopping] SET RECURSIVE_TRIGGERS OFF
GO
ALTER DATABASE [shopping] SET  DISABLE_BROKER
GO
ALTER DATABASE [shopping] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO
ALTER DATABASE [shopping] SET DATE_CORRELATION_OPTIMIZATION OFF
GO
ALTER DATABASE [shopping] SET TRUSTWORTHY OFF
GO
ALTER DATABASE [shopping] SET ALLOW_SNAPSHOT_ISOLATION OFF
GO
ALTER DATABASE [shopping] SET PARAMETERIZATION SIMPLE
GO
ALTER DATABASE [shopping] SET READ_COMMITTED_SNAPSHOT OFF
GO
ALTER DATABASE [shopping] SET HONOR_BROKER_PRIORITY OFF
GO
ALTER DATABASE [shopping] SET RECOVERY FULL
GO
ALTER DATABASE [shopping] SET  MULTI_USER
GO
ALTER DATABASE [shopping] SET PAGE_VERIFY CHECKSUM
GO
ALTER DATABASE [shopping] SET DB_CHAINING OFF
GO
ALTER DATABASE [shopping] SET FILESTREAM( NON_TRANSACTED_ACCESS = OFF )
GO
ALTER DATABASE [shopping] SET TARGET_RECOVERY_TIME = 60 SECONDS
GO
ALTER DATABASE [shopping] SET DELAYED_DURABILITY = DISABLED
GO
EXEC sys.sp_db_vardecimal_storage_format N'shopping', N'ON'
GO
ALTER DATABASE [shopping] SET QUERY_STORE = OFF
```

```sql
GO
USE [shopping]
GO
/****** Object:  Table [dbo].[config]    Script Date: 2020/4/30 21:22:09 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[config](
    [policy_id] [int] NULL,
    [policy] [varchar](50) NULL,
    [value] [varchar](50) NULL
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[customer]    Script Date: 2020/4/30 21:22:09 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[customer](
    [customer_id] [nchar](10) NOT NULL,
    [name] [varchar](50) NOT NULL,
    [grade] [nchar](10) NULL,
    [total_amount] [nchar](10) NULL,
 CONSTRAINT [PK_customer] PRIMARY KEY CLUSTERED
(
    [customer_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[order]    Script Date: 2020/4/30 21:22:09 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[order](
    [order_id] [int] NOT NULL,
    [customer_id] [int] NULL,
    [order_time] [datetime] NULL,
    [amount] [float] NULL,
    [status] [nchar](10) NULL,
 CONSTRAINT [PK_order] PRIMARY KEY CLUSTERED
(
    [order_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

```sql
GO
/****** Object:  StoredProcedure [dbo].[changeStrategy]    Script Date: 2020/4/30 21:22:09 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE proc [dbo].[changeStrategy]
@strategy int
AS
BEGIN
    UPDATE config SET value = @strategy WHERE policy = 'strategy';
    print 'Stratedy updated.'
END
GO
USE [master]
GO
ALTER DATABASE [shopping] SET  READ_WRITE
GO

```