# 实验二 数据库表/视图的创建与维护实验

班级：2015211305 姓名：顾溢杰 学号：2015211257

## 实验目的

1. 通过进行数据库表的建立操作，熟悉并掌握SQL SERVER数据库表的建立方法，理解关系数据库表的结构，巩固SQL标准中关于数据库表的建立语句；
2. 通过进行数据库表数据的增加、删除和插入等维护操作，熟悉并掌握SQL SERVER数据库数据的操作方法，巩固SQL中关于数据维护的语句；
3. 通过对SQL SERVER中建立、维护视图的实验，熟悉SQL SERVER中对视图的操作方法和途径，理解和掌握视图的概念。

## 实验内容

1. 熟悉课程实验背景知识；
2. 根据物理模式使用SQL Server创建相应的表；
3. 将教师提供的数据导入表；
4. 对每个表都尝试增加、删除或者修改一条记录，看看是否能够成功，原因何在？
5. 在SQL SERVER中建立、维护视图。

## 实验环境

两台装有Windows 10操作系统和Microsoft SQL Server 2008 R2的笔记本电脑，一台充当服务器，一台充当客服端。

## 实验步骤、结果及分析

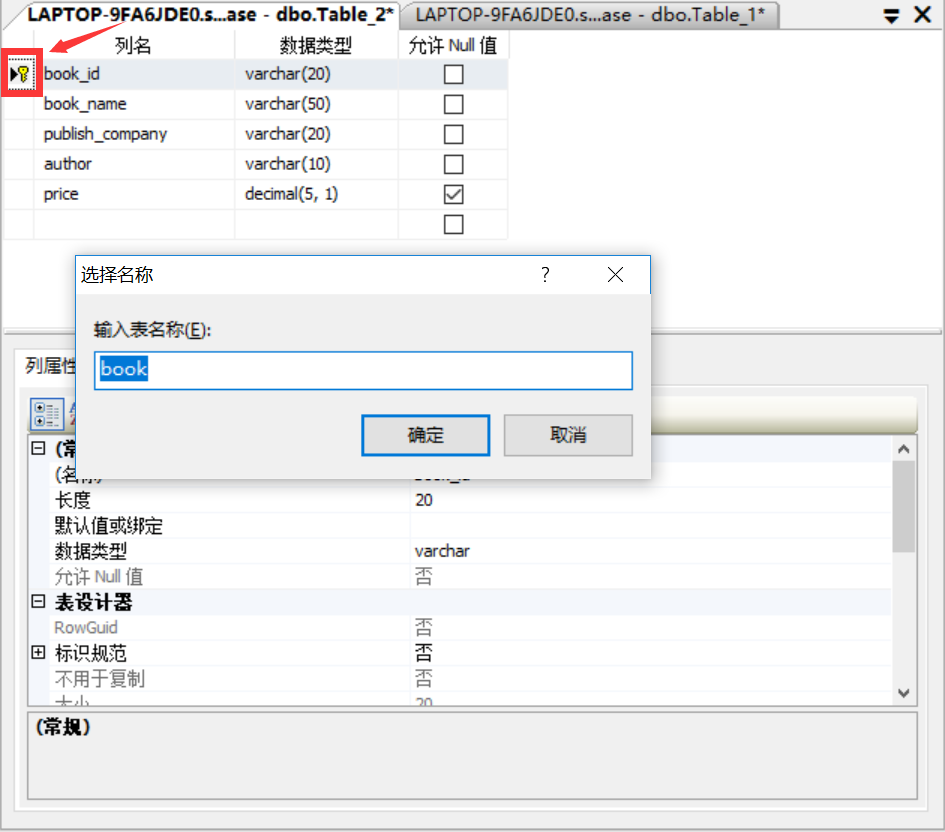
##### 创建数据库和表

实验一中已创建student\_class\_database数据库。接下来创建各表。

1. **book表**

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 是否可以为NULL |
| [主键]book\_id | varchar(20) | 否 |
| book\_name | varchar(50) | 否 |
| publish\_company | varchar(50) | 否 |
| author | varchar(10) | 否 |
| price | decimal(5,1) | 是 |

该表利用图形化界面进行创建。我们在对象资源管理器的student\_class\_database数据库中的表条目上点击右键，选择新建表，弹出新建表的界面，然后填入各列信息。右键点击图中箭头所指取余，选择设置为主键。然后crtl+s保存，输入表名，再点击确定，即可创建成功。图中箭头所指处的图标显示book\_id已经被设置为了主键。



1. **course表**

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 是否可以为NULL |
| [主键]course\_id | varchar(20) | 否 |
| course\_name | varchar(50) | 否 |
| book\_id | varchar(20) | 是 |
| total\_perior | smallint | 否 |
| week\_perior | smallint | 否 |
| credit | tinyint | 是 |

除了要将course\_id设置为主键，还需将book\_id设置为该表相对于book表的外键。

该表利用SQL脚本创建，代码如下：

1. **create** **table** course(
2. course\_id **varchar**(20) not null,
3. course\_name **varchar**(50) not null,
4. book\_id **varchar**(20),
5. total\_perior **smallint** not null,
6. week\_perior **smallint** not null,
7. credit tinyint,
8. **primary** **key**(course\_id)
9. );
10. go
11. **class表**

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 是否可以为NULL |
| [主键]class\_id | varchar(10) | 否 |
| monitor | varchar(10) | 是 |
| classroom | varchar(20) | 否 |
| department\_id | varchar(10) | 否 |

SQL脚本如下：

1. **create** **table** class(
2. class\_id **varchar**(10) not null,
3. monitor **varchar**(10),
4. classroom **varchar**(20) not null,
5. department\_id **varchar**(10) not null,
6. **primary** **key**(class\_id)
7. );
8. go
9. **class\_course表**

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 是否可以为NULL |
| [主键]class\_id | varchar(10) | 否 |
| [主键]course\_id | varchar(20) | 否 |

特别需要注意的是，该表中的主键由class\_id和course\_id共同构成。

SQL脚本如下：

1. **create** **table** class\_course(
2. class\_id **varchar**(10) not null,
3. course\_id **varchar**(20) not null,
4. **primary** **key**(class\_id,course\_id)
5. );
6. go
7. **department表**

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 是否可以为NULL |
| [主键]department\_id | varchar(10) | 否 |
| department\_name | varchar(50) | 否 |
| department\_header | varchar(10) | 是 |
| teacher\_num | smallint | 是 |

SQL脚本如下：

1. **create** **table** department(
2. department\_id **varchar**(10) not null,
3. department\_name **varchar**(50) not null,
4. department\_header **varchar**(10),
5. teacher\_num **smallint**,
6. **primary** **key**(department\_id)
7. );
8. go
9. **student表**

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 是否可以为NULL |
| [主键]student\_id | varchar(10) | 否 |
| student\_name | varchar(10) | 否 |
| sex | varchar(2) | 否 |
| birth | date | 是 |
| class\_id | varchar(10) | 否 |
| entrance\_date | date | 否 |
| home\_addr | varchar(200) | 是 |

SQL脚本如下：

1. **create** **table** student(
2. student\_id **varchar**(10) not null,
3. student\_name **varchar**(10) not null,
4. sex **varchar**(2) not null,
5. birth **date**,
6. class\_id **varchar**(10) not null,
7. entrance\_date **date** not null,
8. home\_addr **varchar**(200),
9. **primary** **key**(student\_id)
10. );
11. go
12. **student\_course表**

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 是否可以为NULL |
| [主键]course\_id | varchar(20) | 否 |
| [主键]student\_id | varchar(10) | 否 |
| grade | tinyint | 是 |
| credit | tinyint | 否 |
| semester | tinyint | 否 |
| school\_year | varchar(10) | 否 |

SQL脚本如下：

1. **create** **table** student\_course(
2. course\_id **varchar**(20) not null,
3. student\_id **varchar**(10) not null,
4. grade tinyint,
5. credit tinyint not null,
6. semester tinyint not null,
7. school\_year **varchar**(10) not null,
8. **primary** **key**(course\_id,student\_id)
9. );
10. go
11. **teacher表**

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 是否可以为NULL |
| [主键]teacher\_id | varchar(10) | 否 |
| teacher\_name | varchar(10) | 否 |
| sex | varchar(2) | 否 |
| birth | date | 是 |
| department\_id | varchar(10) | 否 |
| profession | varchar(10) | 否 |
| telephone | varchar(15) | 否 |
| home\_addr | varchar(200) | 是 |
| postalcode | varchar(10) | 是 |

SQL脚本如下：

1. **create** **table** teacher(
2. teacher\_id **varchar**(10) not null,
3. teacher\_name **varchar**(10) not null,
4. sex **varchar**(2) not null,
5. birth **date**,
6. department\_id **varchar**(10) not null,
7. profession **varchar**(10) not null,
8. telephone **varchar**(15) not null,
9. home\_addr **varchar**(200),
10. postalcode **varchar**(10),
11. **primary** **key**(teacher\_id)
12. );
13. go
14. **teacher\_course\_class表**

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 是否可以为NULL |
| [主键]teacher\_id | varchar(10) | 否 |
| [主键]course\_id | varchar(20) | 否 |
| [主键]class\_id | varchar(10) | 否 |
| semester | tinyint | 否 |
| school\_year | varchar(10) | 否 |
| course\_schedule | varchar(10) | 否 |
| course\_classroom | varchar(10) | 否 |
| book\_id | varchar(20) | 是 |

SQL脚本如下：

1. **create** **table** teacher\_course\_class(
2. teacher\_id **varchar**(10) not null,
3. course\_id **varchar**(20) not null,
4. class\_id **varchar**(10) not null,
5. semester tinyint not null,
6. school\_year **varchar**(10) not null,
7. course\_schedule **varchar**(10) not null,
8. course\_classroom **varchar**(10) not null,
9. book\_id **varchar**(20),
10. **primary** **key**(teacher\_id,course\_id,class\_id)
11. );
12. go

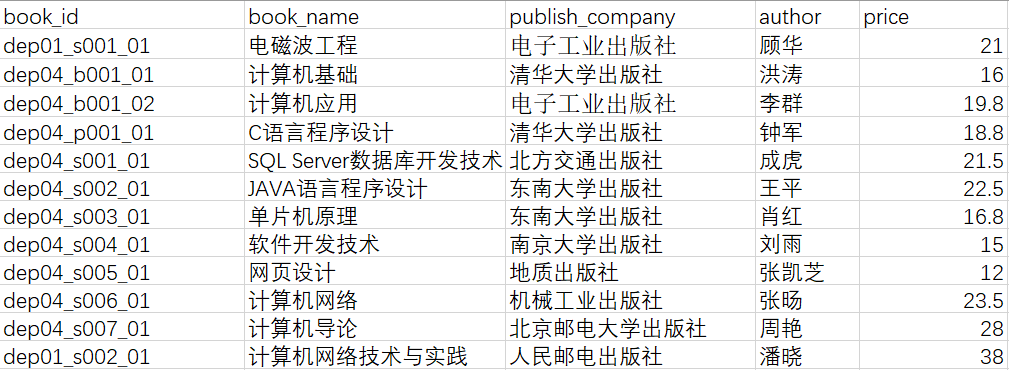
##### 导入表数据

SQL提供的导入数据的方法很多。可以通过SQL语句一条记录、一条记录的插入；也可以通过图形化表格界面键入；还可以直接从EXCEL导入一个表（在数据库中新建一个表或者添加到数据库中已有的表中）；也可以通过SQL语句，从EXCEL中将数据插入到已有的表中。我们将采用第四种方法。

老师提供的表中，所有的表混合在了一个sheet中，因此我们先将其拆分成9个sheet，或者9个EXCEL文件。



例如book.xls内部数据如图所示



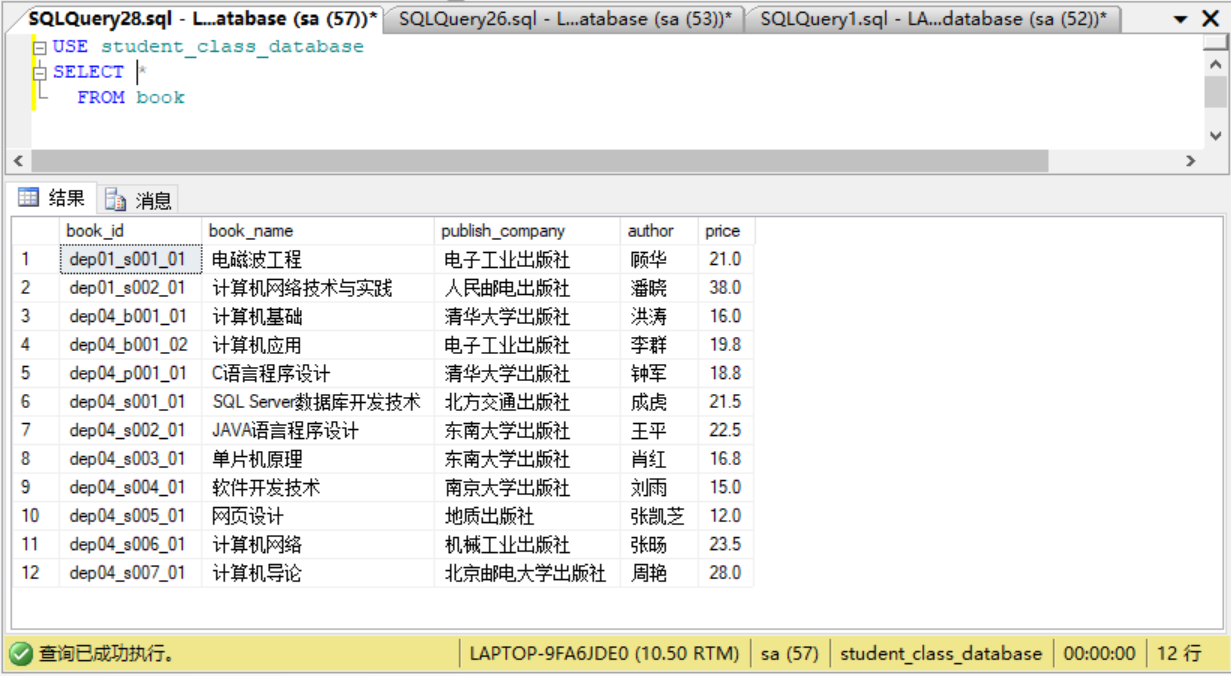
启用Ad Hoc Distributed Queries

1. exec sp\_configure 'show advanced options',1
2. reconfigure
3. exec sp\_configure 'Ad Hoc Distributed Queries',1
4. reconfigure

从book.xls导入数据到book表中

1. INSERT INTO book
2. SELECT \*
3. FROM OPENROWSET('MICROSOFT.JET.OLEDB.4.0', 'EXCEL 5.0;HDR=YES;DATABASE=C:\Users\16417\Desktop\notation\database\courses\实验\实验二\数据\book.xls', sheet1$)

编写SQL脚本查询book表中数据，如下图，表中数据无误，导入成功。



其余表格用相同方法导入，不再赘述。

##### 数据库表数据的增加、删除、修改

**本步骤中，前四个表的操作在本机上进行；后五个表的操作通过客户端远程连接到本机进行。首先介绍远程连接的相关配置。**

远程连接配置

1. 在sql server配置管理器中启动sql server browser服务。如果无法启动，到windows 10的服务管理中启用（为了方便，选择自动启动）。
2. 在windows防火墙设置中开放本机的1433端口。在入站规则中添加1433端口即可。
3. 在windows防火墙例外中添加sqlservr.exe程序（在SQL Server的安装目录中）
4. 重启sql server中的相关服务或者重启服务器主机。
5. 保证服务器主机网络通畅后，在客户端主机上远程连接。输入服务器主机的ip地址，选择SQL Server身份验证，输入用户名和密码即可登陆成功。

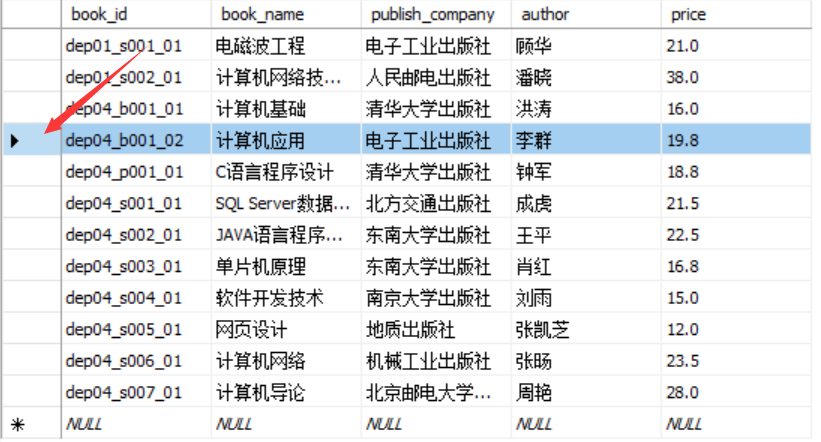
**接下来是在本机进行的前四个表的操作。**

book表

采用图形化界面操作进行表内记录的增加、删除、修改。

1. **删除**

在对象资源管理器中选中book表，右键单击选择编辑前200行。



在上图所示的编辑界面中，点击最左边的空白区域即可选中某一行，右键单击，选择删除，即可删除该行的记录。也可以选中多行。



从这图可见，《计算机应用》这本书的记录已经成功被我们删除。

1. **增加**

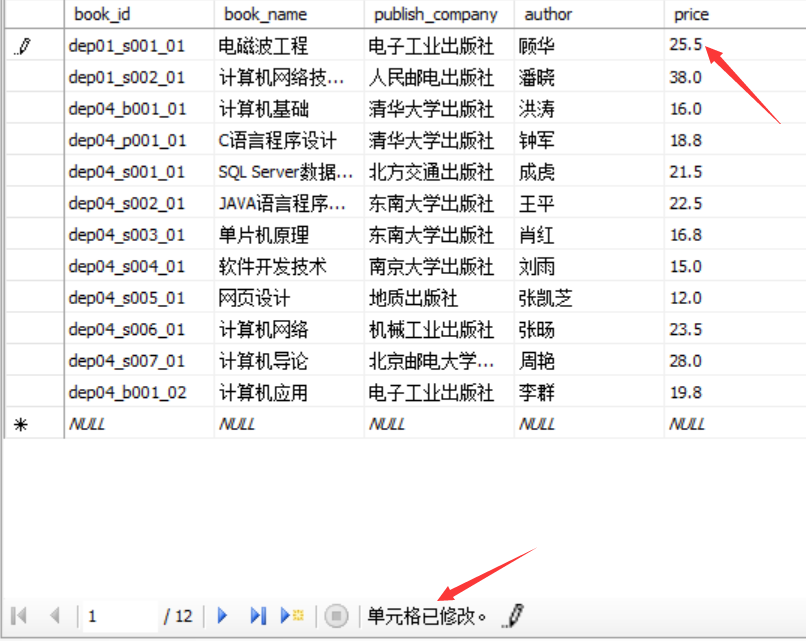
接下来我们在下面显示为NULL的行中将该记录添加回去。



从上图可见，该记录被我们成功添加了回去。

1. **修改**

我们也可以修改某一条记录，例如，将电磁波工程这本书的价格改为25.5



course表

采用SQL脚本。

1. 增加

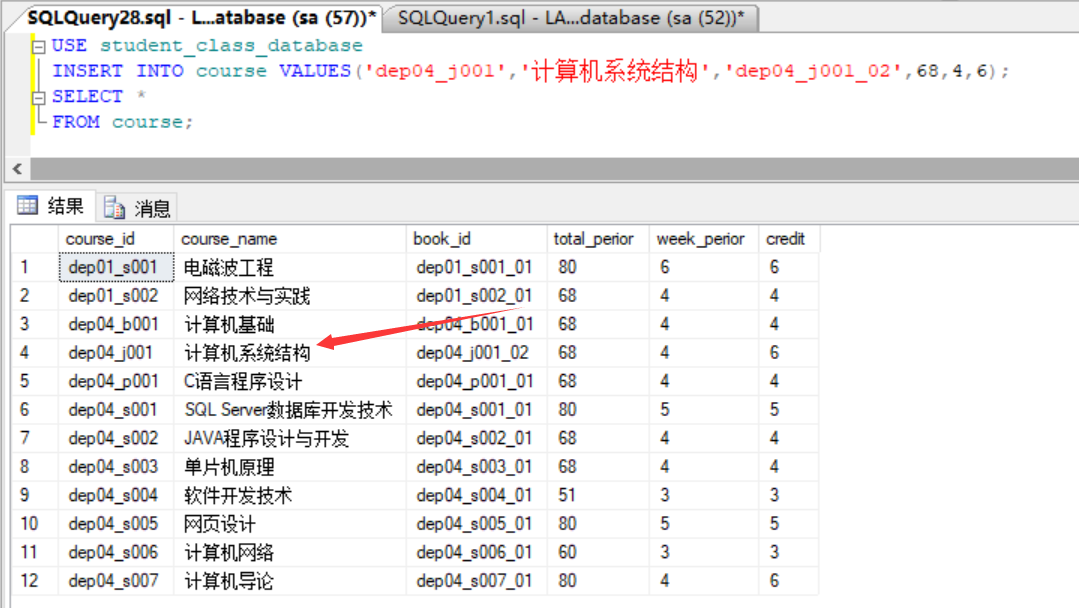
我们向course表中增加一条记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| course\_id | course\_name | book\_id | total\_perior | week\_perior | credit |
| dep04\_j001 | 计算机系统结构 | dep04\_j001\_01 | 68 | 4 | 6 |

SQL脚本如下：

1. USE student\_class\_database
2. INSERT INTO course VALUES('dep04\_j001','计算机系统结构','dep04\_j001\_02',68,4,6);
3. SELECT \*
4. FROM course;

结果如下：



1. 删除

我们再将计算机系统结构这本书删除，以其course\_id作为筛选条件。

SQL脚本如下：

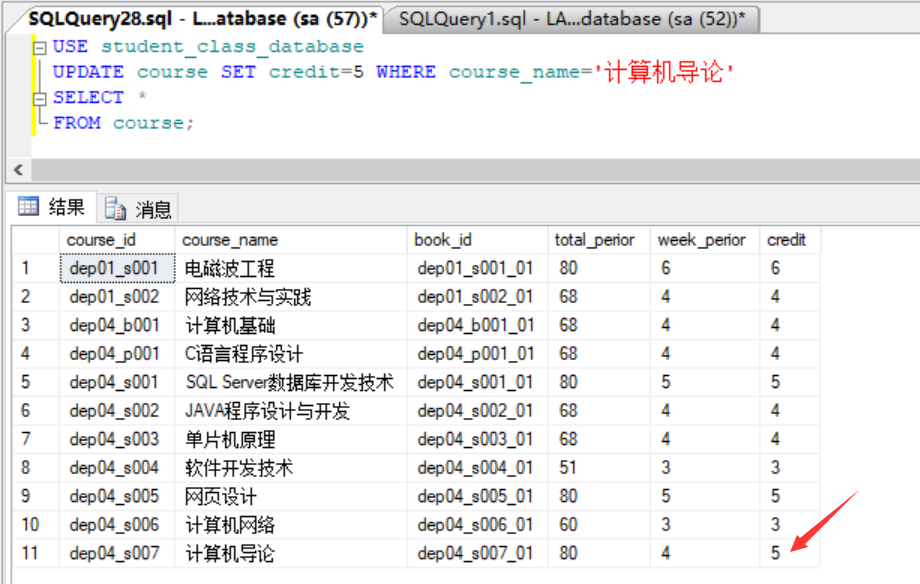
1. USE student\_class\_database
2. DELETE FROM course WHERE course\_id='dep04\_j001';
3. SELECT \*
4. FROM course;
5. 修改

我们修改计算机导论的credit属性取值为5（原先为6），采用course\_name作为筛选条件。

SQL脚本如下：

1. USE student\_class\_database
2. UPDATE course SET credit=5 WHERE course\_name='计算机导论'
3. SELECT \*
4. FROM course;

结果如下：



class表

1. 增加

我们向class表中增加一条记录

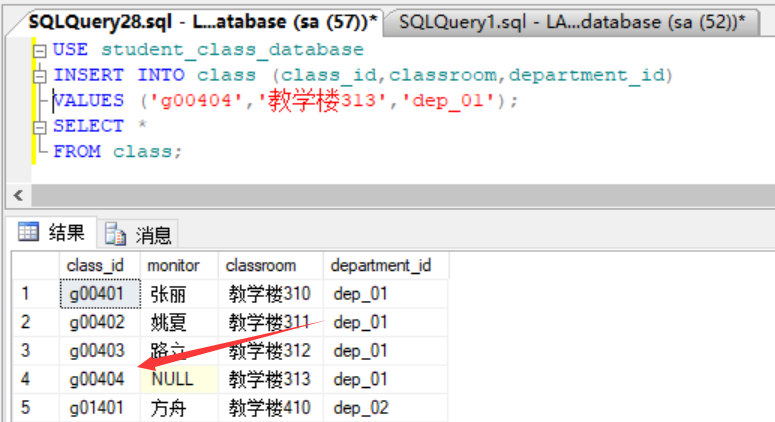
|  |  |  |  |
| --- | --- | --- | --- |
| class\_id | monitor | classroom | department\_id |
| g00404 | NULL | 教学楼313 | dep\_01 |

之前我们在新建表的时候，设定了monitor属性取值可以为NULL。（可能新建了班级，但是还未选举出班长）

SQL脚本如下：

1. USE student\_class\_database
2. INSERT INTO **class** (class\_id,classroom,department\_id)
3. VALUES ('g00404','教学楼313','dep\_01');
4. SELECT \*
5. FROM **class**;

结果如下：



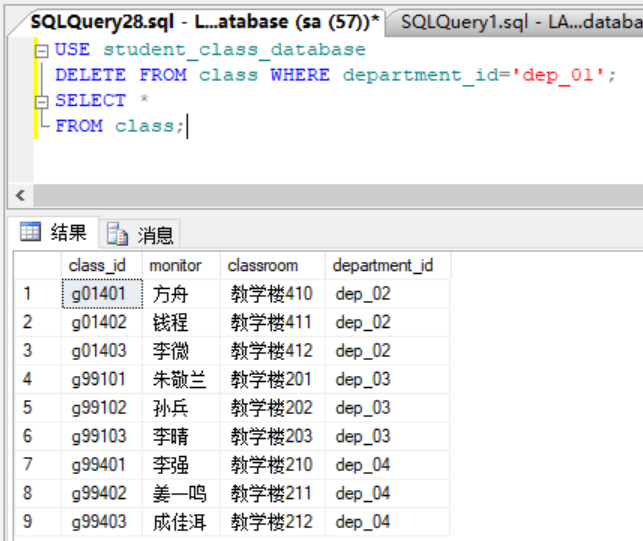
1. 删除

删除department\_id为dep\_01的所有班级记录

SQL脚本如下：

1. USE student\_class\_database
2. DELETE FROM **class** WHERE department\_id='dep\_01';
3. SELECT \*
4. FROM **class**;

结果如下：



1. 修改

我们将class\_id为g00401的班级的monitor修改为’李娜’，class\_id作为筛选条件

SQL脚本如下：

1. USE student\_class\_database
2. UPDATE **class** SET monitor='李娜' WHERE class\_id='g00401'
3. SELECT \*
4. FROM **class**
5. WHERE class\_id='g00401';

结果如下：



class\_course表

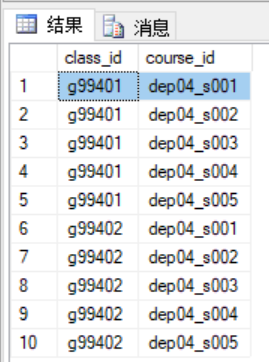
采用SQL脚本。

增加class\_id为g99401的课程记录，要求所有课程都和班级g99402一样；然后删除class\_id为g99402的所有记录；最后将class\_id为g99403的记录的class\_id改为g99402。

SQL脚本如下：

1. USE student\_class\_database
2. INSERT INTO class\_course(class\_id,course\_id)(
3. SELECT 'g99401',T.course\_id
4. FROM class\_course AS T
5. WHERE T.class\_id='g99402'
6. );
8. DELETE FROM class\_course
9. WHERE class\_id='g99402';
11. UPDATE class\_course
12. SET class\_id='g99402'
13. WHERE class\_id='g99403'

结果如下：



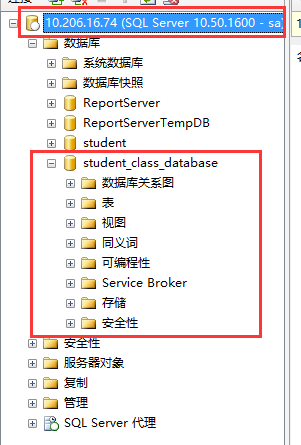
**接下来是在客服端主机的远程操作。**

department表

首先在客户端主机上远程连接到服务器主机。



如下图，成功连接后，在客户端主机的Management studio的对象资源管理器中，可以看到我们在服务器主机上创建的数据库student\_course\_database。



接下来我们便可以远程操作该数据库了。

修改department\_header为’王敬远’的记录的teacher\_num为100；

删除department\_name为通信与信息工程的记录；

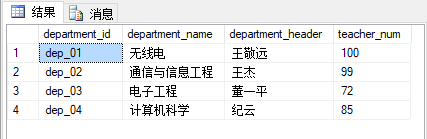
新增如下记录：

|  |  |  |  |
| --- | --- | --- | --- |
| department\_id | department\_name | department\_header | teacher\_num |
| dep\_02 | 通信与信息工程 | 王杰 | 99 |

SQL脚本如下：

1. use student\_class\_database
3. UPDATE department SET teacher\_num=100
4. WHERE department\_header='王敬远';
6. DELETE FROM department
7. WHERE department\_name='通信与信息工程'
9. INSERT INTO department
10. VALUES('dep\_02','通信与信息工程','王杰',99);
12. select \*
13. from department;

结果如下：



student表

采用SQL脚本。

新增一条记录如下：

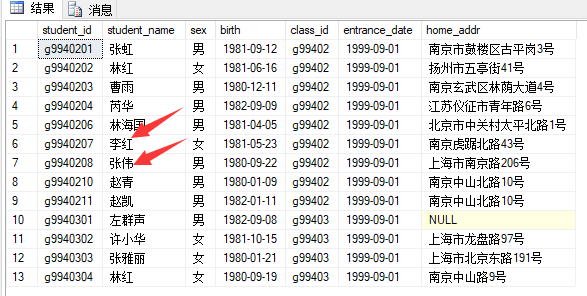
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| student\_id | student\_name | sex | birth | class\_id | entrance\_date | home\_addr |
| g9940208 | 张伟 | 男 | 1980-09-22 | g990402 | 1999-09-01 | 上海市南京路206号 |

然后将student\_name为李红[大]的记录删除，将student\_name为李红[小]的记录的student\_name改为李红。

SQL脚本如下：

1. use student\_class\_database
3. INSERT INTO student
4. VALUES('g9940208','张伟','男','1980-09-22',
5. 'g99402','1999-09-01','上海市南京路206号');
7. DELETE FROM student WHERE student\_name='李红[大]';
9. UPDATE student SET student\_name='李红'
10. WHERE student\_name='李红[小]';
12. select \*
13. from student;

结果如下：



student\_course表

新增张伟（student\_id=g9940208）的一条选课记录如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| course\_id | student\_id | grade | credit | semester | school\_year |
| dep04\_b001 | g9940208 | NULL | 4 | 0 | 1999/2000 |

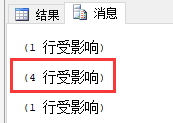
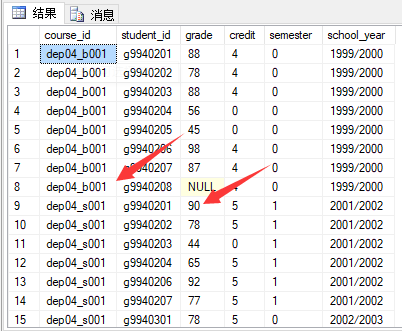
删除student\_id为g9940205的学期1的选课记录；

修改student\_id为g9940201的course\_id为dep04\_s001的课程成绩为90。

SQL脚本如下：

1. use student\_class\_database
3. INSERT INTO student\_course
4. VALUES('dep04\_b001','g9940208',NULL,4,0,'1999/2000');
6. DELETE FROM student\_course
7. WHERE student\_id='g9940205' and semester = 1;
9. UPDATE student\_course SET grade=90
10. WHERE student\_id='g9940201' and course\_id='dep04\_s001';
12. select \*
13. from student\_course;

结果如下：



说明student\_id为g9940205的学期1的选课记录有4条，均被删除。

teacher表

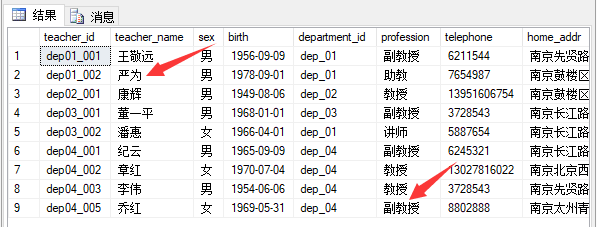
修改teacher\_id为dep04\_005的profession为副教授；删除teacher\_id为’dep04\_004’的记录；新增一条记录如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| teacher\_id | teacher\_name | sex | birth | department\_id |
| dep01\_002 | 严为 | 男 | 1978-09-01 | dep\_01 |
| profession | telephone | home\_addr | postalcode |  |
| 助教 | 7654987 | 南京鼓楼区长虹路3号 | 210002 |  |

SQL脚本如下：

1. use student\_class\_database
3. UPDATE teacher SET profession='副教授'
4. WHERE teacher\_id='dep04\_005';
6. DELETE FROM teacher
7. WHERE teacher\_id='dep04\_004';
9. INSERT INTO teacher
10. VALUES('dep01\_002','严为','男','1978/09/01',
11. 'dep\_01','助教','7654987','南京鼓楼区长虹路3号','210002');
13. select \*
14. from teacher

结果如下：



teacher\_course\_class表

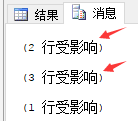
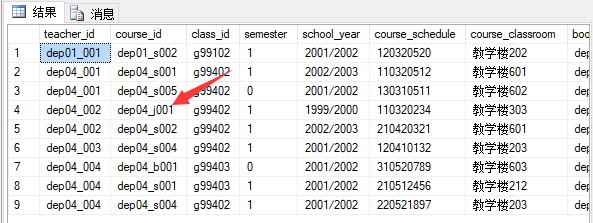
将教学楼211的记录的上课教室改为教学楼203；删除school\_year为1999/2000的所有记录；新增一条记录如下：

|  |  |  |  |
| --- | --- | --- | --- |
| teacher\_id | course\_id | class\_id | semester |
| dep04\_002 | dep04\_j001 | g99402 | 1 |
| school\_year | course\_schedule | course\_classroom | book\_id |
| 1999/2000 | 110320234 | 教学楼303 | dep04\_j001\_01 |

SQL脚本如下：

1. use student\_class\_database
3. UPDATE teacher\_course\_class
4. SET course\_classroom='教学楼203'
5. WHERE course\_classroom='教学楼211';
7. DELETE FROM teacher\_course\_class
8. WHERE school\_year='1999/2000';
10. INSERT INTO teacher\_course\_class VALUES('dep04\_002','dep04\_j001','g99402',1,'1999/2000','110320234','教学楼303','dep04\_j001\_01');
12. select \*
13. from teacher\_course\_class

结果如下：



可见有上课教室为教学楼211的记录有两条，均被改为了教学楼203；school\_year为1999/2000的记录有三条，均被删除；最后还剩一条新添加进去的记录的school\_year为1999/2000。

##### 建立、维护视图

在进行视图的实验之前，为了消除上述实验的影响，我们先将数据库恢复到初始状态。

什么是视图？

在 SQL 中，视图是基于 SQL 语句的结果集的可视化的表。

视图包含行和列，就像一个真实的表。视图中的字段就是来自一个或多个数据库中的真实的表中的字段。我们可以向视图添加 SQL 函数、WHERE 以及 JOIN 语句，我们也可以提交数据，就像这些来自于某个单一的表。

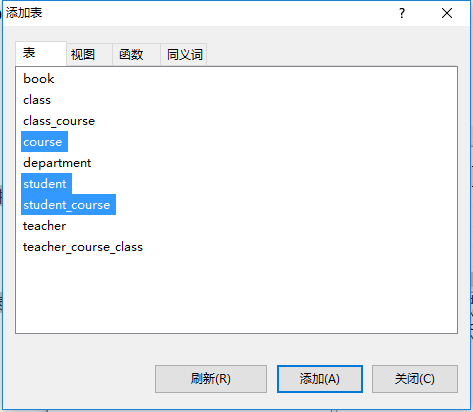
注释：数据库的设计和结构不会受到视图中的函数、where 或 join 语句的影响。

利用图形化界面创建视图

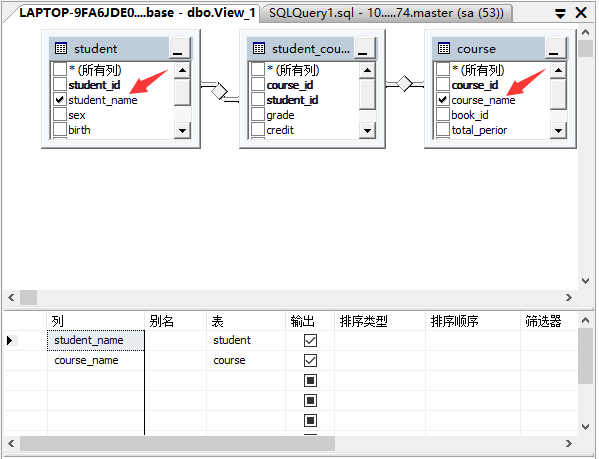
* 在 “对象资源管理器”中，展开要创建新视图的数据库。
* 右键单击“视图”文件夹，然后单击“新建视图...”。
* 在 “添加表” 对话框中，从以下选项卡之一选择要在新视图中包含的元素：“表”、“视图”、“函数”和“同义词”。
* 单击 “添加”，再单击 “关闭”。
* 在 “关系图窗格”中，选择要在新视图中包含的列或其他元素。
* 在 “条件窗格”中，选择列的其他排序或筛选条件。
* 在“文件”菜单上，单击“保存\*\*\*视图名称”。
* 在 “选择名称” 对话框中，输入新视图的名称并单击 “确定”。

我们想要构造一个有学生姓名和选课名称的视图。

首先选择student、student\_course、course表：

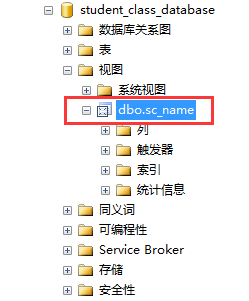


然后在视图的设计界面选择student表的student\_name属性和course表的course\_name属性。



点击保存，取名为sc\_name。

可以看到对象资源管理器中多出了dbo.sc\_name这个视图。

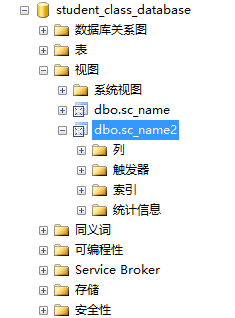


利用SQL脚本创建视图

SQL脚本如下：

1. USE student\_class\_database;
2. GO
3. CREATE VIEW sc\_name2
4. AS
5. SELECT s.student\_name, c.course\_name
6. FROM student as s,student\_course as sc,course as c
7. WHERE s.student\_id=sc.student\_id and sc.course\_id=c.course\_id;
8. GO

结果如下：



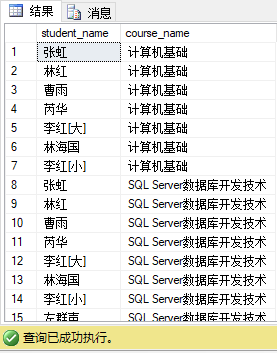
可以看到又多出了一个db.sc\_name2的视图，其本质上与sc\_name视图的功能一样。

查询视图

我们输出之前创建的视图的所有列，SQL脚本如下：

1. USE student\_class\_database;
2. SELECT \*
3. FROM sc\_name2

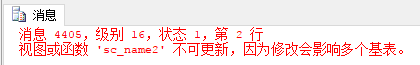
结果如下：



共44条记录，与student\_course表中的记录相符。

向视图中增加数据

我们试图向该视图中增加一条(‘李娜’,’计算机系统结构’)的记录。结果如下：



可见无法在这种涉及多个基表的视图中添加记录。

我们另外再创建一个较简单的视图teacher\_simple，该试图是teacher表的简化版本，只包含那些可以不为NULL的属性。

SQL脚本如下：

1. USE student\_class\_database;
2. GO
3. CREATE VIEW teacher\_simple
4. AS
5. SELECT teacher\_id,teacher\_name,sex,department\_id,profession,telephone
6. FROM teacher

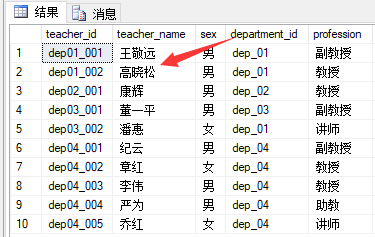
我们在视图中新增一条记录如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| teacher\_id | teacher\_name | sex | department\_id | profession | telephone |
| dep01\_002 | 高晓松 | 男 | dep\_01 | 教授 | 123456789 |

SQL脚本如下：

1. USE student\_class\_database;
2. INSERT INTO teacher\_simple VALUES('dep01\_002','高晓松','男','dep\_01','教授','123456789')
3. SELECT \*
4. FROM teacher\_simple

结果如下：

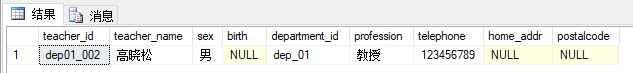


我们再去teacher表中查询检验

SQL脚本如下：

1. USE student\_class\_database;
2. SELECT \*
3. FROM teacher
4. WHERE teacher\_name='高晓松'

结果如下：



可见对视图中记录的修改，也会影响到基表。

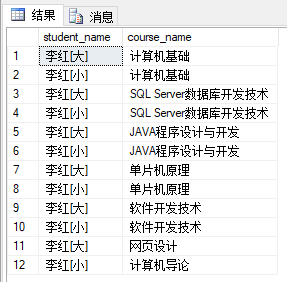
修改视图

我们将之前创建的sc\_name2视图修改为姓李的学生的姓名和选课名字。

SQL脚本如下：

1. ALTER VIEW sc\_name2
2. AS
3. SELECT s.student\_name, c.course\_name
4. FROM student as s,student\_course as sc,course as c
5. WHERE s.student\_id=sc.student\_id and sc.course\_id=c.course\_id and s.student\_name like '李%';
6. GO
7. SELECT \*
8. FROM sc\_name2

结果如下：



## 实验总结

通过本实验，我加深理解了数据库表的结构，掌握了建立数据库表的方法，熟悉了SQL中关于数据库表的建立语句；掌握了从外部将数据导入到SQL Server数据库中的若干种方法；掌握了远程连接、操作数据库的方法；通过对每个数据表的增加、删除和插入记录的维护操作，熟悉并掌握了SQL Server数据库数据的操作方法；最后通过建立、维护视图，初步理解和掌握了视图的概念。

当然也碰到了不少问题，例如在导入数据时，发现excel中多余列虽然是空的，但是SQL Server仍认为这其中有数据，这就导致了导入时属性个数不匹配，因此需要将处理excel文件；

碰到了两个问题，一个是excel包含隐藏列（虽然没有数据但是还是被认为有数据），导致导入时列数对不上。另一个是属性超出预定义的属性范围，导致导入出错。

# 实验一 SQL SERVER安装、数据库创建与维护实验

班级：2015211305 姓名：顾溢杰 学号：2015211257

## 实验目的

1．通过对SQL SERVER2000的安装和简单使用：

（1）了解安装SQL SERVER2000的软硬件环境和安装方法；

（2）熟悉SQL SERVER2000相关使用；

（3）熟悉SQL SERVER2000的构成和相关工具；

（4）通过SQL SERVER2000的使用来理解数据库系统的基本概念。

2．通过创建数据库、并进行相应的维护，了解并掌握SQL SERVER数据库的创建和维护的不同方法和途径，并通过这一具体的数据库理解实际数据库所包含的各要素。

## 实验平台及环境

* Thinkpad x1 carbon 6th 笔记本
* Windows 10操作系统
* Microsoft SQL Server 2008 R2 企业版 32位

## 实验内容

1．SQL SERVER2008 安装

（1） 在Windows 10上安装并运行SQL SERVER2008。

（2） 练习启动和停止数据库服务。

（3） 通过SQL SERVER Enterprise Manager连接数据库。

（4） 熟悉SQL SERVER Enterprise Manager的各项功能，了解SQL SERVER的主要对象。

(5) 了解SQL SERVER在安装时自动创建的数据库和几类系统表。

2．数据库创建与维护

1. 创建“学生选课”数据库，要求有相应的日志文件；
2. 对数据库属性和参数进行查询、相应的修改和维护，内容包括：

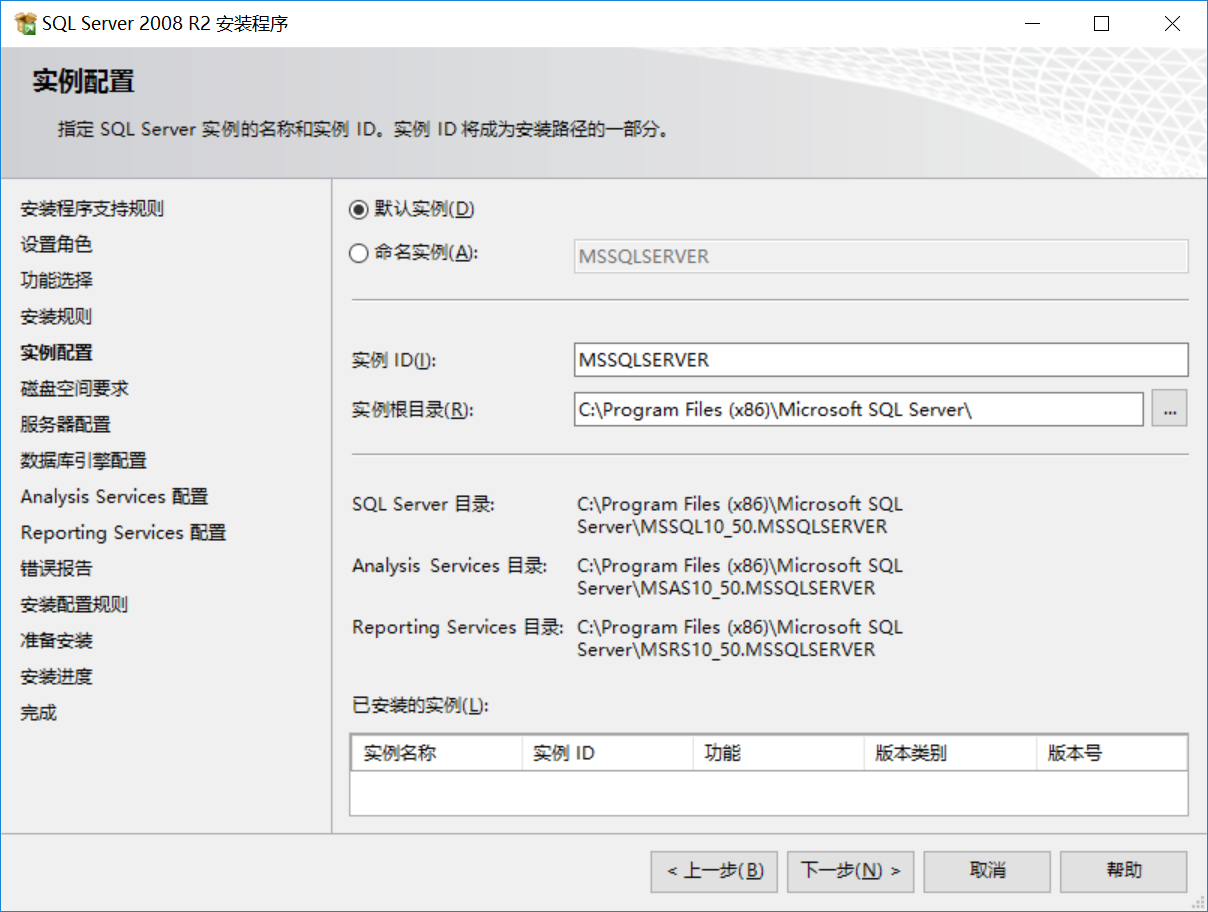
调整数据库的大小，完成数据库大小的增加、减小；修改日志文件的最大值；查看数据库的属性值；

1. 练习数据库的删除等维护；
2. 用Enterprise Manager管理工具和交互式的Transact\_SQL语句分别完成以上操作。

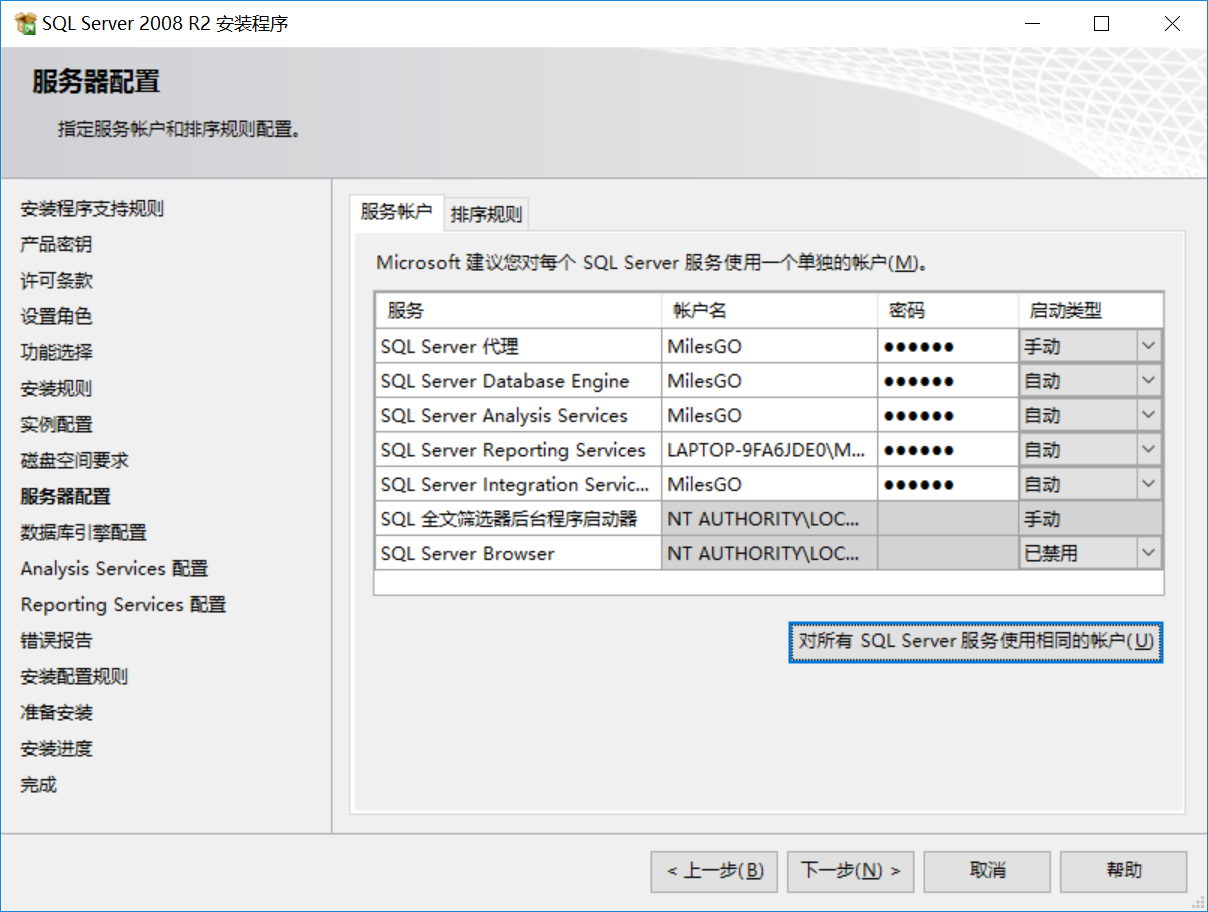
## 实验步骤、结果及分析

##### SQL Server 2008 R2安装

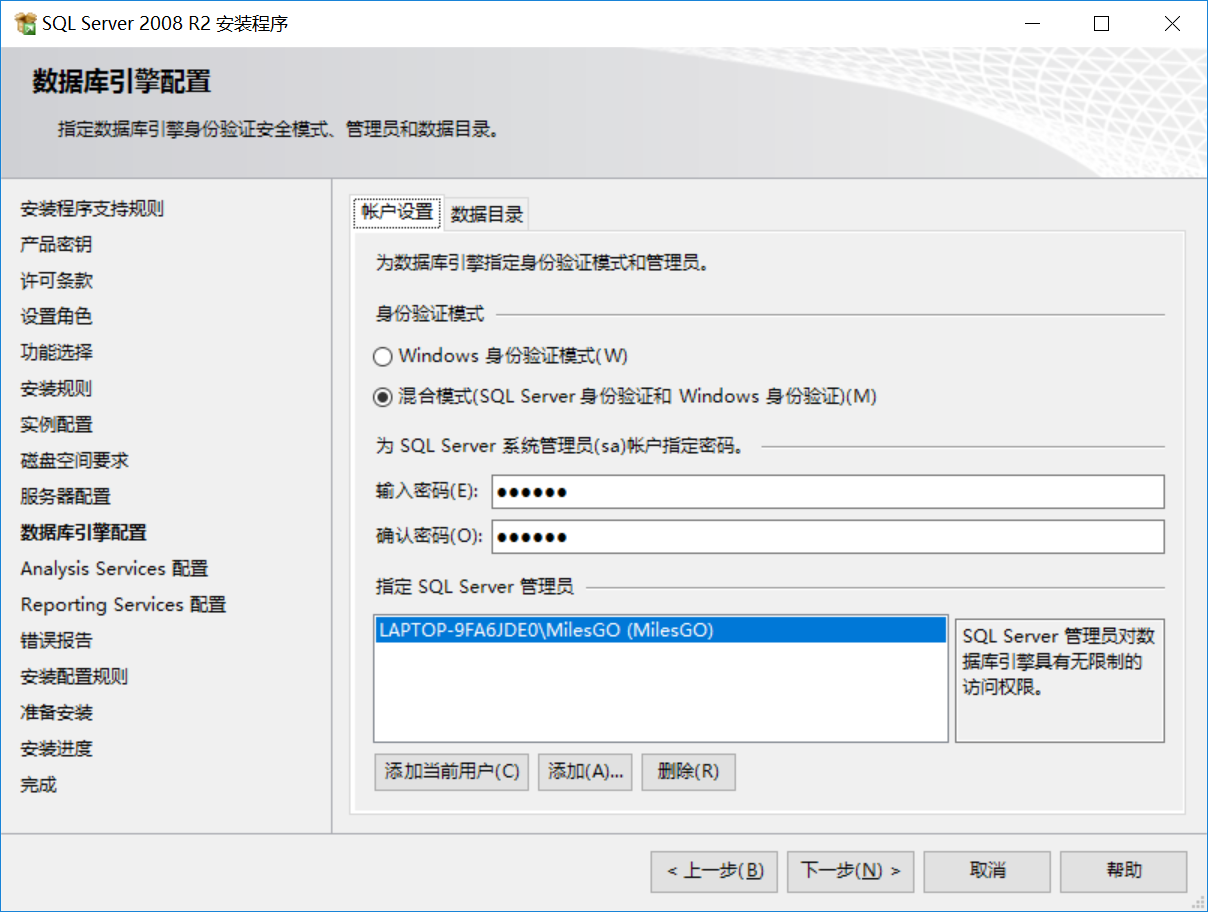
参照《SQLSERVER2008 R2安装说明》，进行了实例配置、服务器配置、服务器引擎配置等，添加了本机账户为SQL Server管理员，最终在Windows 10 操作系统上成功安装。



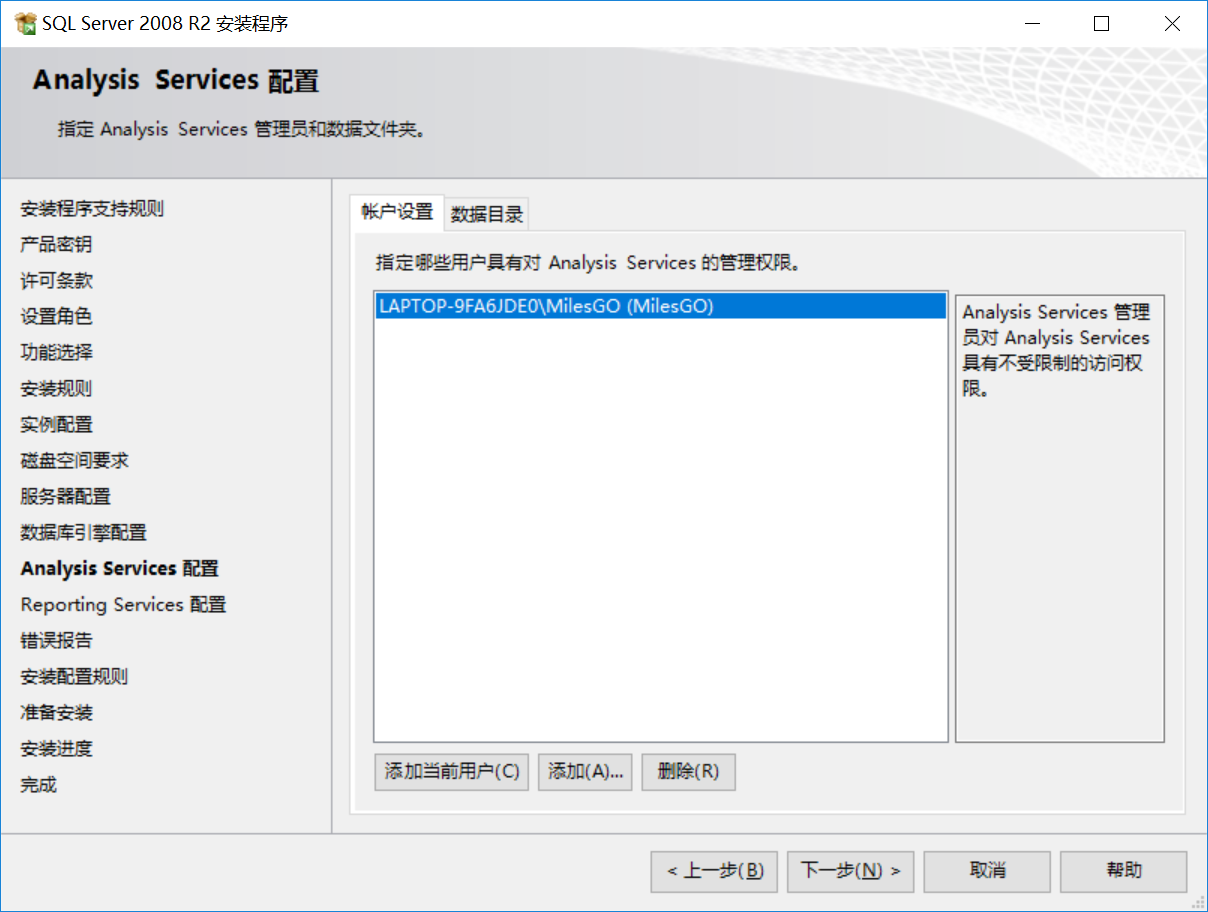
实例配置



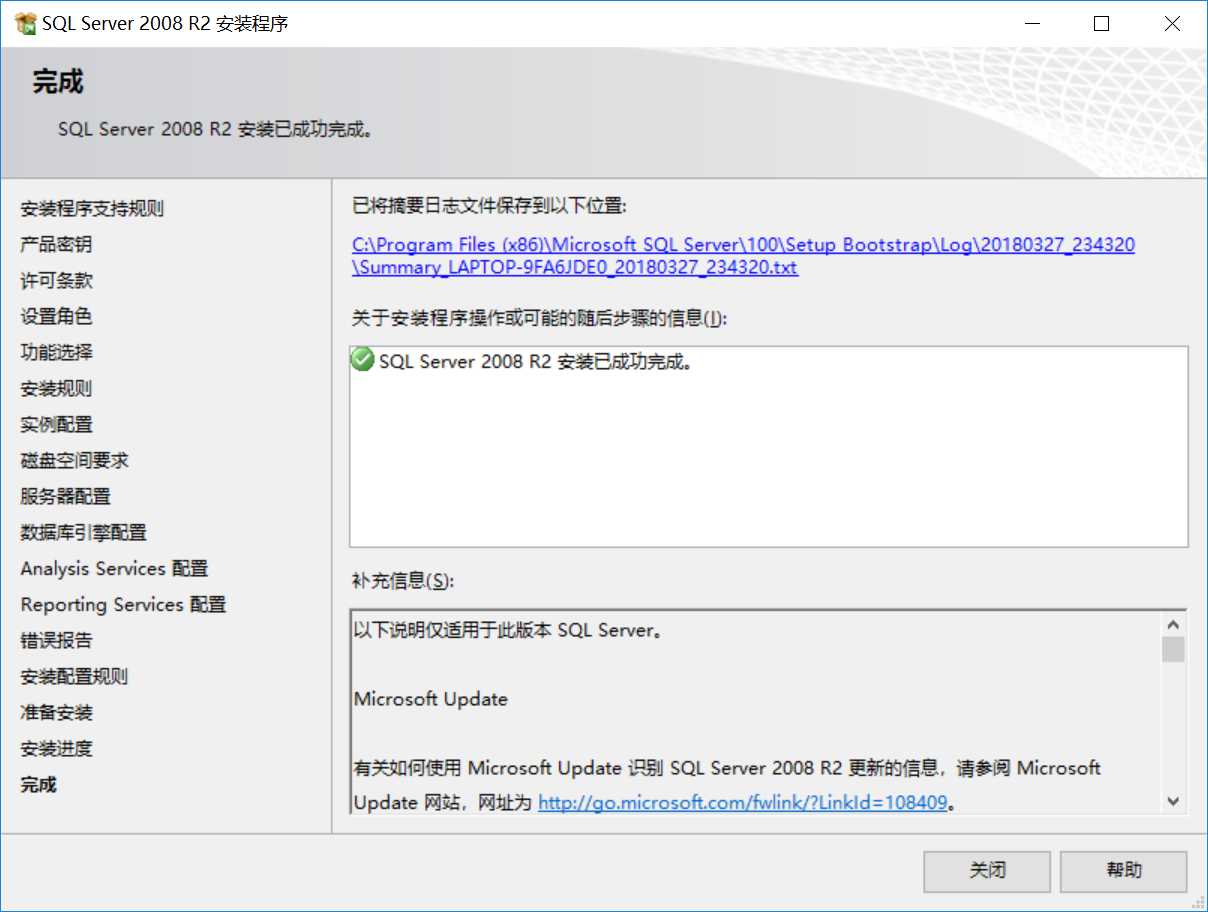
服务器（账户）配置



数据库引擎（账户）配置



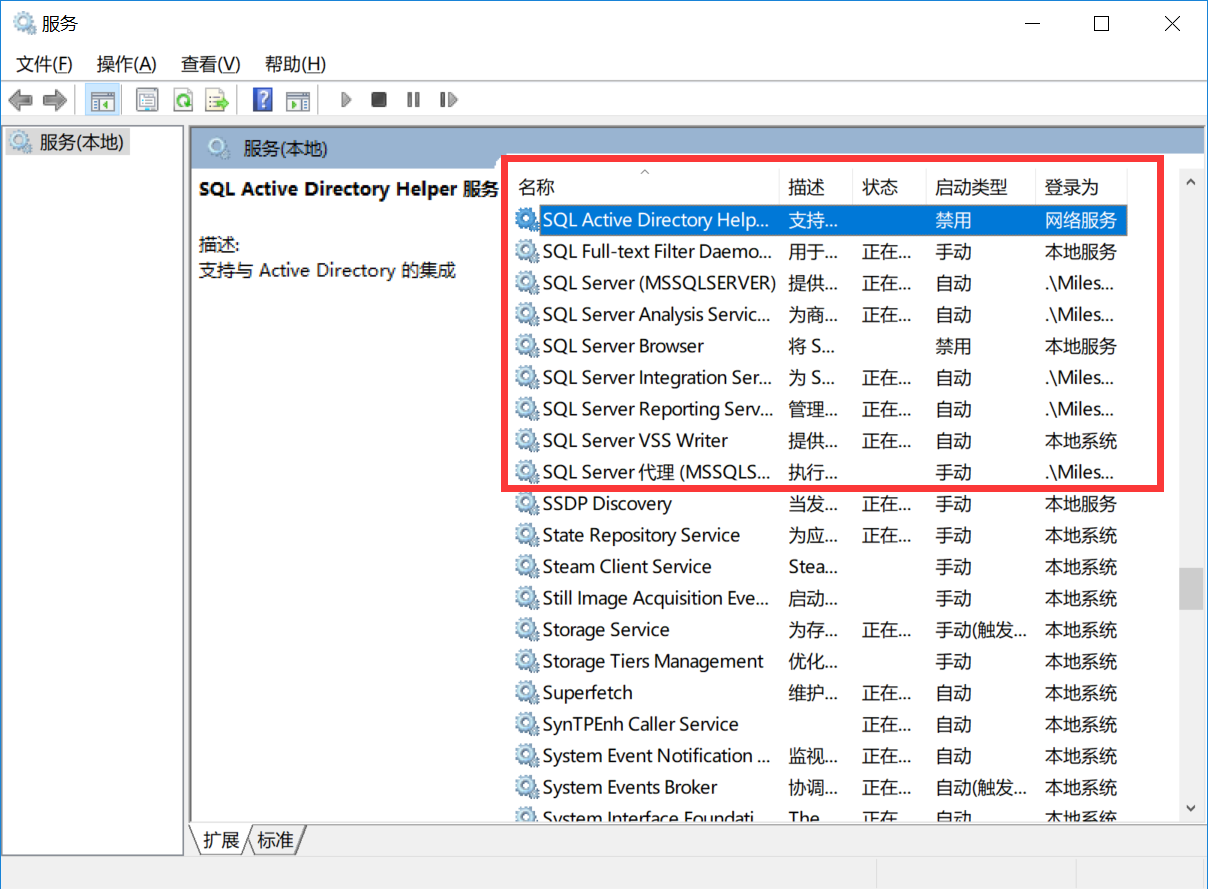
Analysis Services 账户配置



安装完成

##### 启动和停止数据库服务

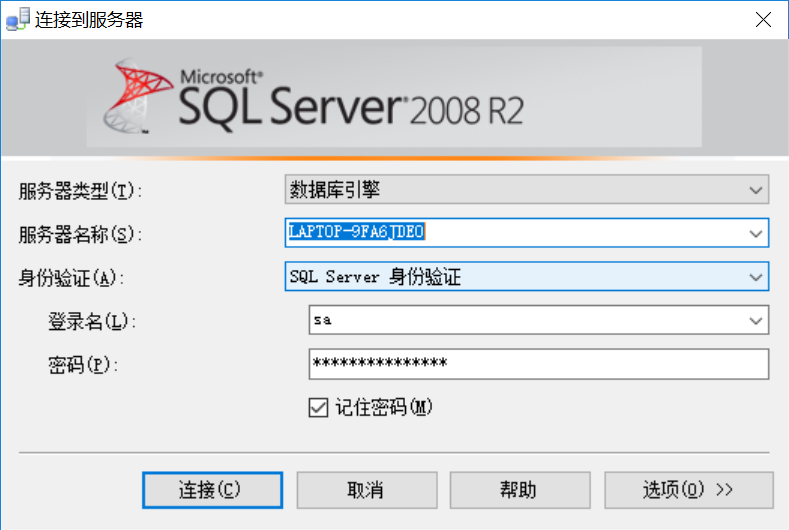
打开控制面板-管理工具-服务，按名称排序，找到与SQL Server相关的数据库服务条目，右键选择停止或启动。为了后续实验的顺利进行，最后保持为启动状态。



##### 连接数据库

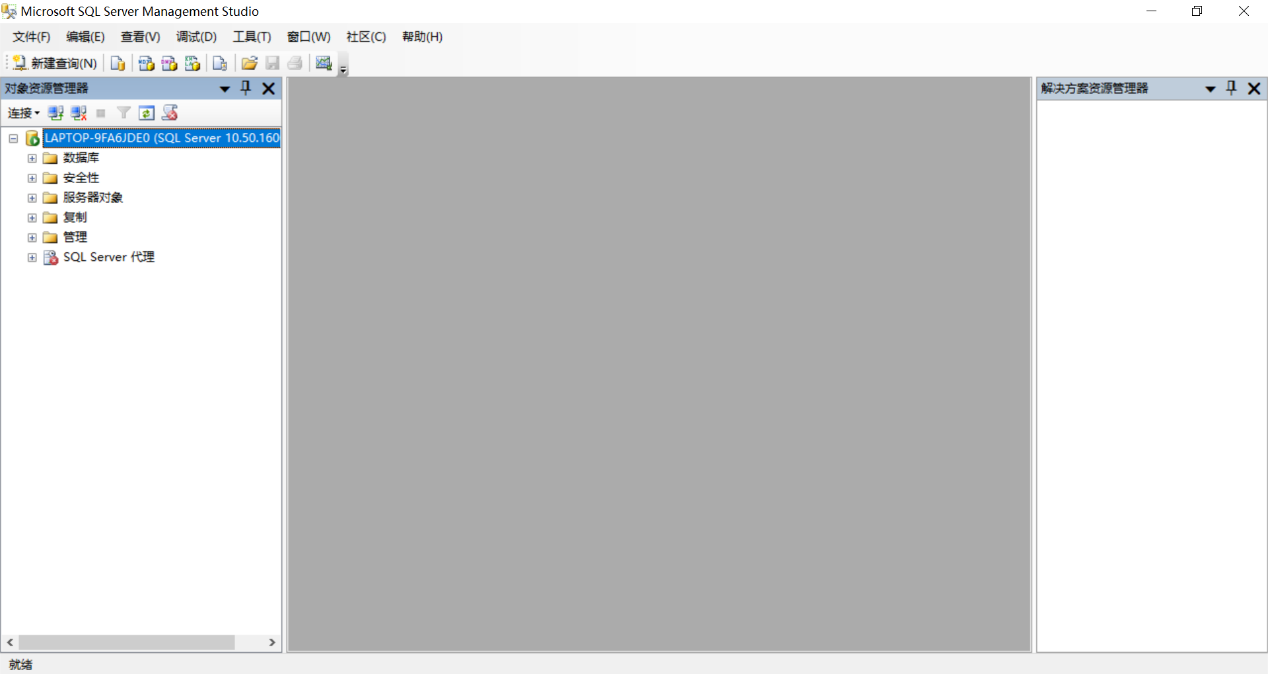
点击打开Microsoft SQL Server Management Studio，显示连接到服务器的登录界面，选择

SQL Server身份验证，输入安装时设置的用户名和密码，点击连接，如下图所示。

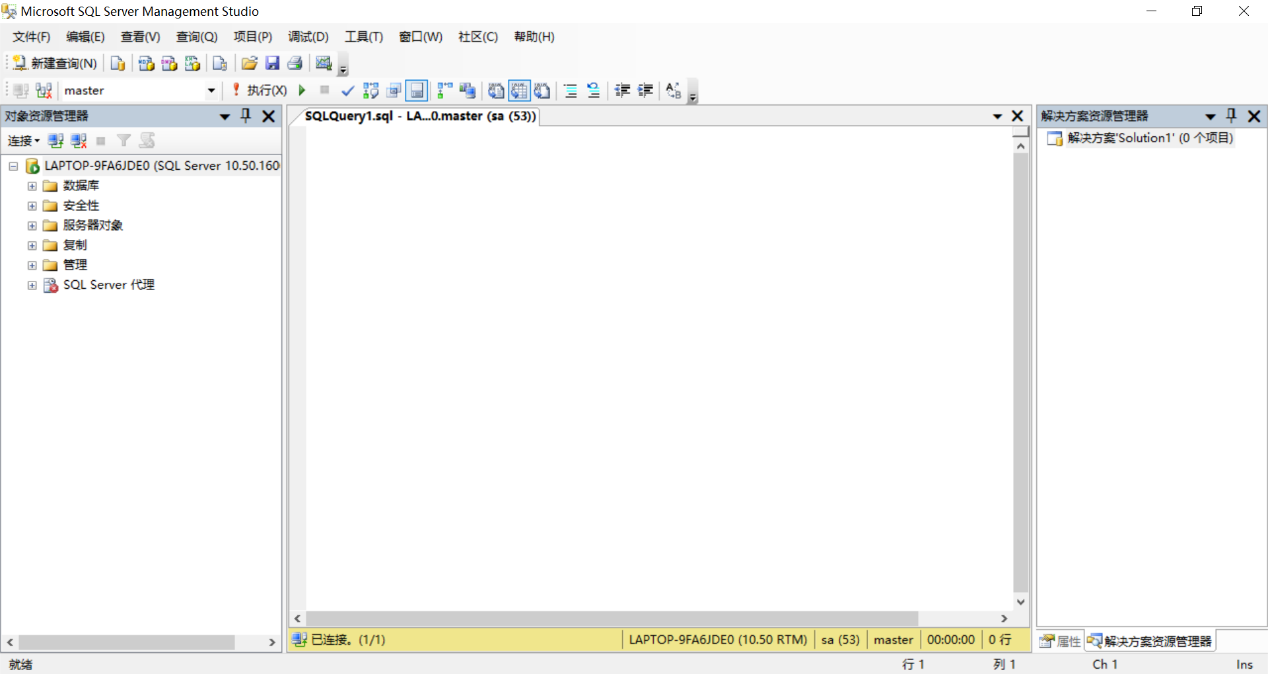


##### 熟悉SQL SERVER Management Studio的各项功能，了解SQL SERVER的主要对象

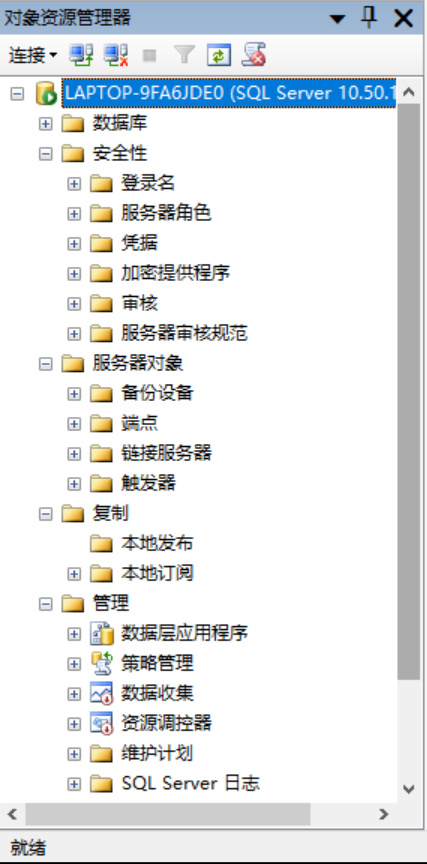
下面是Management Studio的主界面，左边是对象资源管理器，右边是解决方案资源管理器。



点击左上角的新建查询，中间的灰色区域便会编程SQL语句脚本的编辑界面。



观察对象资源管理器，可以初步了解SQL Server中有哪些对象，例如数据库、安全性相关对象、服务器对象等等。

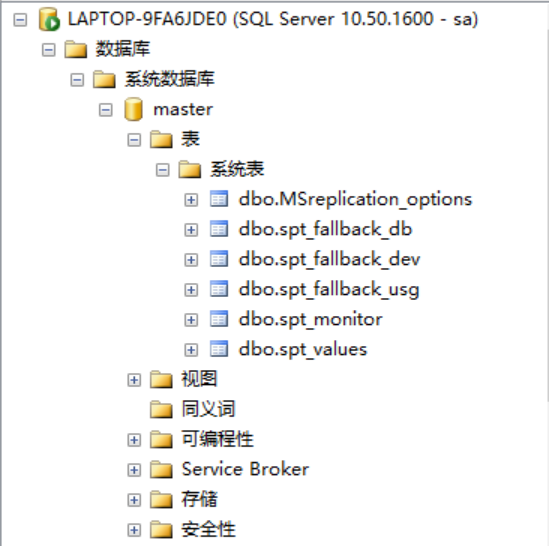


##### 了解SQL SERVER在安装时自动创建的数据库和几类系统表

在左边的对象资源管理器中打开数据库分支，便可以看到SQL Server在安装时自动创建的数据库。有master、model、msdb和tempdb，下面首先介绍这几个自动创建的数据库。

* master  
  master数据库是SQL Server系统最重要的数据库，它记录了SQL Server系统的所有系统信息。这些系统信息包括所有的登录信息、系统设置信息、SQL Server的初始化信息和其他系统数据库及用户数据库的相关信息。因此，如果 master 数据库不可用，则 SQL Server 无法启动。
* model  
  model 数据库用作在 SQL Server 实例上创建的所有数据库的模板。因为每次启动 SQL Server 时都会创建 tempdb数据库，所以 model 数据库必须始终存在于 SQL Server 系统中。当发出 CREATE DATABASE（创建数据库）语句时，将通过复制 model 数据库中的内容来创建数据库的第一部分，然后用空页填充新数据库的剩余部分。 如果修改 model 数据库，之后创建的所有数据库都将继承这些修改。例如，可以设置权限或数据库选项或者添加对象，例如，表、函数或存储过程。
* msdb  
  msdb数据库是代理服务数据库，为其报警、任务调度和记录操作员的操作提供存储空间。
* tempdb  
  tempdb是一个临时数据库，它为所有的临时表、临时存储过程及其他临时操作提供存储空间。tempdb数据库由整个系统的所有数据库使用，不管用户使用哪个数据库，他们所建立的所有临时表和存储过程都存储在tempdb上。SQL Server每次启动时，tempdb数据库被重新建立。当用户与SQL Server断开连接时，其临时表和存储过程自动被删除。

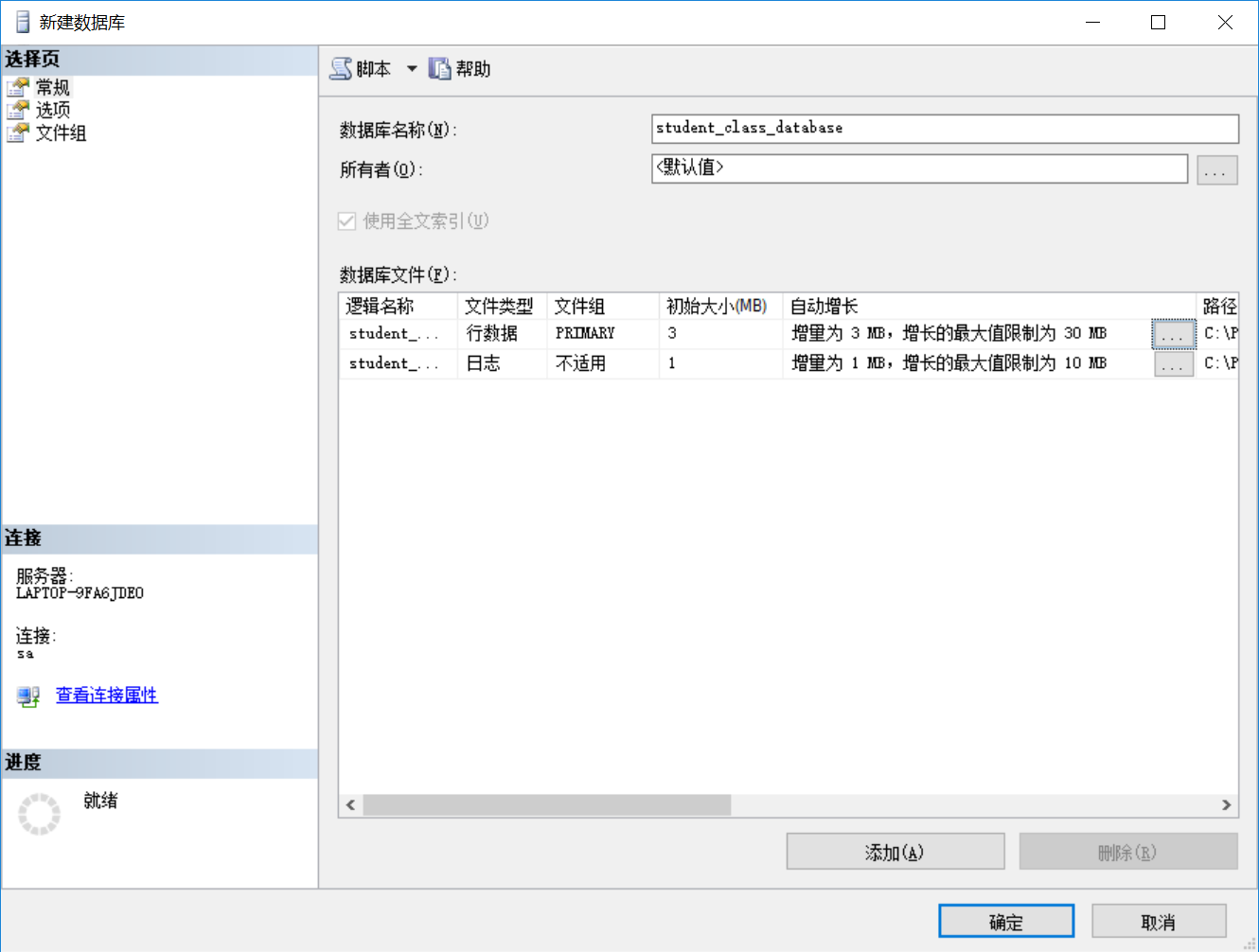
点开这些数据库的表项，再点开系统表，就可以看到这些数据库内置的系统表。例如master数据库的系统表如下：



##### 创建“学生选课”数据库

图形化界面操作

在对象资源管理器的数据库条目上点击右键选择新建数据库，如下图所示。数据库名称为student\_class\_database，采用默认设置，创建了存储行数据的student\_class.mdf，设置为PRIMARY主文件，初始大小为3MB，每次增量为3MB，上限为30MB；另外创建了存储日志的student\_class\_log.ldf，初始大小为1MB，每次增量为1MB，上限为10MB。

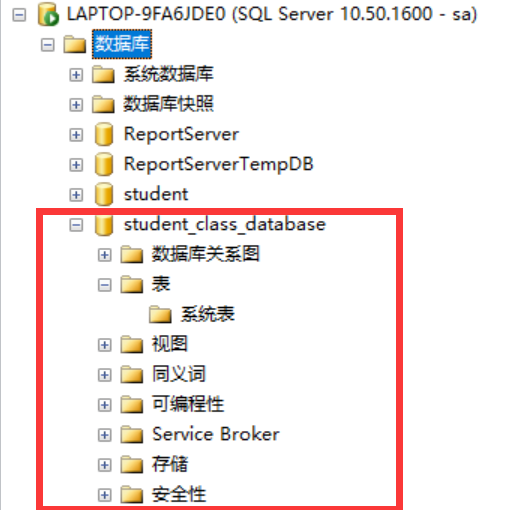


SQL脚本

1. **create** **database** student\_class\_database
2. **on** **primary**
3. (
4. **name**=student\_class\_data,    //存放数据的主文件
5. filename='C:\Users\16417\Desktop\SQL\student\_class\_database.mdf',
6. **size**=3mb,    //文件初始大小为3MB
7. maxsize=30mb,    //文件大小上限为30MB
8. filegrowth=3mb    //文件每次增量为3MB
9. )
10. log **on**
11. (
12. **name**=student\_class\_log,    //日志文件
13. filename='C:\Users\16417\Desktop\SQL\student\_class\_database\_log.ldf',
14. **size**=1mb,
15. maxsize=10mb,
16. filegrowth=1mb
17. );

文件的大小、增量及上限设置与图形化操作方法相同，不再赘述。

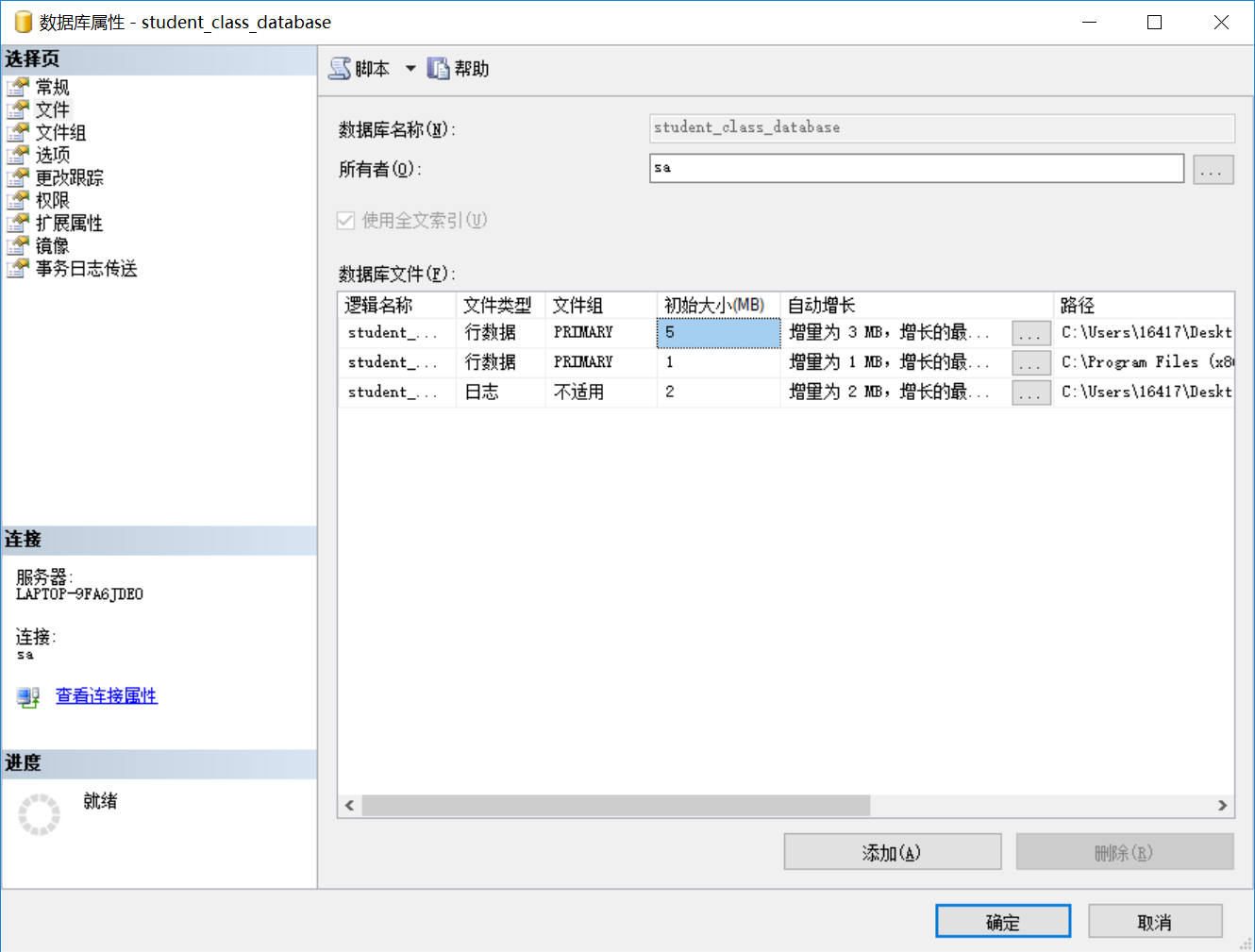
在数据库条目上右键再点击刷新后，可以看到对象资源管理器的数据库中多出了student\_class\_database这一项。



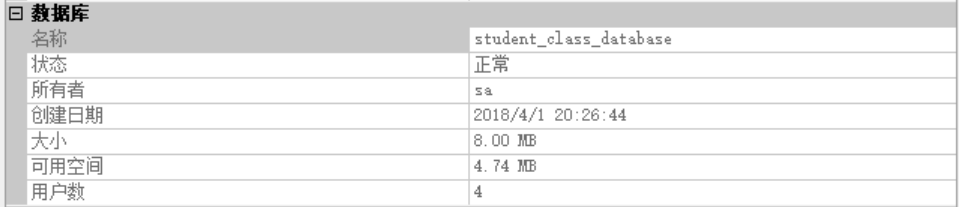
##### 调整数据库的大小

图形化界面操作

在对象资源管理器中，选中student\_class\_database，右键点击选择属性，在打开的属性设置界面的左上角，点击选择页一栏中的文件选项卡，就显示出了文件相关的属性设置页面，我们可以对文件的初始大小进行修改，也可以修改增量和上限。注意增量不能大于文件大小的上限。如下图所示，将主文件的初始大小修改为了5MB，并添加了一个新的文件，命名为student\_class\_data2.ndf，初始大小为1MB，增量为1MB，上限为10MB。最后，将日志文件的初始大小调整为了2MB，最大值调整为了20MB，增量调整为了2MB。



点击确定后，我们再重新进入属性界面，查看常规属性，如下图。



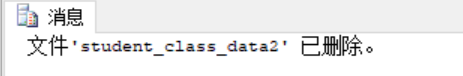
大小为8MB，我们所创建的数据库的mdf、ndf、ldf三个文件加起来，确实等于8MB。

SQL脚本

首先，我们尝试删除student\_class\_data2.ndf这个文件，删除文件的前提条件是文件为空。我们没有往数据库中添加任何数据，所以该文件必然是空的。

1. **ALTER** **DATABASE** student\_class\_database
2. REMOVE FILE student\_class\_data2;

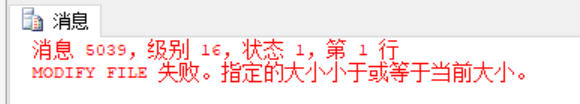
返回了删除成功的消息，如下图。



然后我们尝试增加student\_class\_data.mdf的初始大小为6MB。

1. **ALTER** **DATABASE** student\_class\_database
2. **MODIFY** FILE
3. (**NAME** = student\_class\_data,
4. **SIZE** = 6MB);

接着尝试减小student\_class\_data.mdf的初始大小为3MB

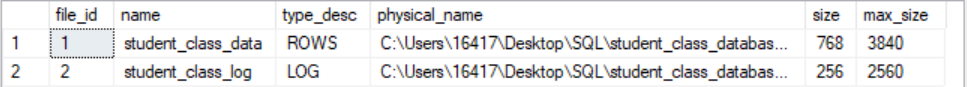


缩小失败，可见无法用该SQL语句缩小文件，而只能增大文件。

在经过上述操作后，我们尝试查看student\_class\_database的文件信息。

1. USE student\_class\_database;
2. GO
3. **SELECT** file\_id, **name**, type\_desc, physical\_name, **size**, max\_size
4. **FROM** sys.database\_files ;

结果如下图所示

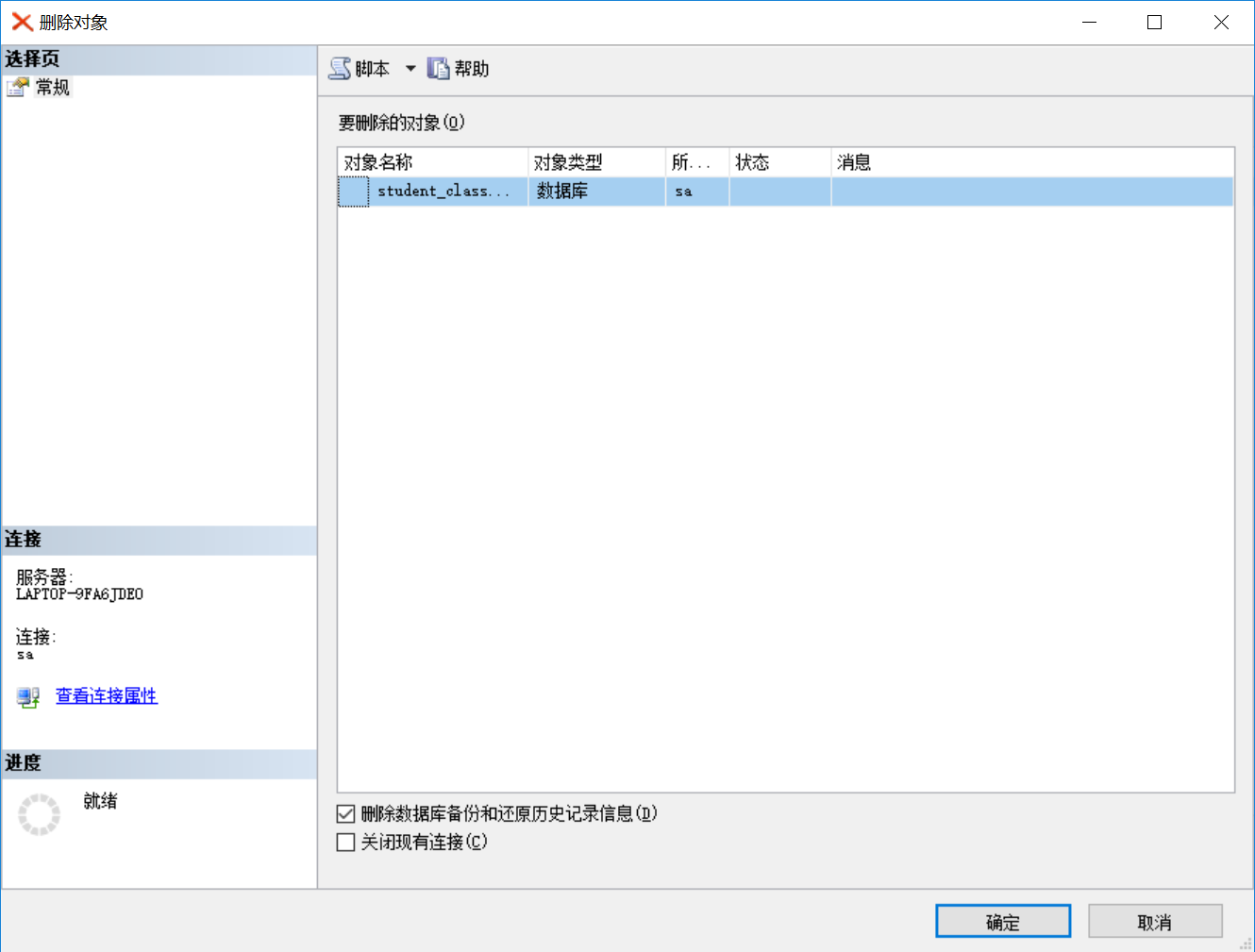


其中size和max\_size属性的单位是8KB，即student\_class\_data的文件大小为6MB，上限为30MB，student\_class\_log的文件大小为2MB，上限为20MB，与操作的结果相符合。

##### 数据库的删除

图形化界面操作

只需在左边的对象资源管理器中，找到student\_class\_database数据库，右键点击，选择删除，弹出如下界面

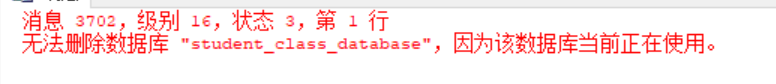


点击确定即可删除数据库。

SQL脚本

1. **DROP** **DATABASE** student\_class\_database

结果显示删除失败



原因是当前处于使用该数据的状态，我们只需要将利用USE语句，将使用状态转移到别的数据库，例如master，即可顺利的删除当前的数据库。

1. USE master
2. GO
3. **DROP** **DATABASE** student\_class\_database

这样就成功删除了。

## 实验总结

在本次实验中，我成功安装了Microsoft SQL Server 2008 R2 企业版，学会了数据库服务的启动与停止，熟悉了SQL Server Management Studio软件，了解了4个”数据库的数据库”，即数据库在安装时就自动创建的一些数据库。学会了在其图形化界面上进行操作，实现了数据库的创建，文件的添加、删除、修改，数据库的删除等操作。同时也了解了基本的SQL语句，编写SQL脚本实现了上述功能。

本次实验为后续的数据库实验和学习奠定了基础。