南京工业大学

《数据库原理与应用》
上机实验指导书

（地理信息系统专业用）

南京工业大学测绘学院

目　录

上机守则

1. 学生必须按指导教师安排的上机实验时间进入机房上机，未经许可，不得带外人进入机房。
2. 进入机房时必须穿上鞋套，否则不得进入机房。
3. 认真填写上机情况登记表，若遇计算机有异常情况，应先向老师汇报，不得擅自处理。
4. 遵守计算机操作规程，即开机时先开显示器再开主机；结束时须关闭计算机，关机时先通过 Windows 功能关闭系统，主机电源指示灯灭了以后再关闭显示器。
5. 禁止上机时玩游戏或从事与上机实验无关的内容。
6. 保持机房安静和整洁，不得携带食品、饮料进入机房，严禁随地吐痰、乱扔垃圾或杂物，禁止吸烟、大声喧哗或闲聊。
7. 爱护机房设施，严禁更改设置参数、添加口令或删除非本人文件。对于导致计算机不能正常工作、影响他人上机者，将取消其上机资格。
8. 严禁私自拆卸配件或将室内物品带出室外。一经发现，除要求按价赔偿外，将通报批评和取消其上机资格，情节严重者交有关行政部门和司法部门处理。

上机实验 1　数据库基本操作

1 .1 实验目的

1、掌握使用对象资源管理器创建数据库；
2、掌握使用 T-SQL 语句创建和修改数据库；
3、练习调用系统存储过程数据库选项、设置数据库选项。

1.2 实验练习预备知识点

1 .2 .1 数据库的存储结构

1 、SQL Server 中创建的数据库的存储结构：包括数据文件和事务日志文件。

2 、数据文件：用于实际存储数据、索引等数据库对象的文件。分为主数据文件（ .mdf ）和非主数据文件（.ndf）。一个数据库可以设置一个或多个数据文件，只能有一个且必须有一个主数据文件。

3 、事务日志文件（ .ldf ）：用来记录用户对数据库进行的所有操作，是维护数据库完整性的重要工具。一个数据库可以设置一个或多个事务日志文件。

1.2.2 数据文件的基本属性

1、文件名 （ NAME ）：指定该数据文件的文件名；
2、 位置 （ FILENAME ）：指定存放该数据文件的目录；包含路径和文件名。如： D:\SQL SERVER\mos_data.MDF 。
3、初始大小 （ SIZE ）：该数据文件的初始容量。默认是 1MB 。
4、文件组 （ FILEGROUP ）：该数据文件所属的文件组。默认文件组是 PRIMARY 文件组。 也可以让数据文件属于其他文件组，但需要先新建文件组。
5、文件增长方式 ：选中文件自动增长，则数据文件根据需要自动增长。有两种增长方式：
① 按兆字节增长：指定每次增长的兆字节数； ② 按百分比增长：指定每次增长的百分比。不选文件自动增长，数据文件大小是固定的。
6、最大文件大小 （ MAXSIZE ）：设置数据库文件的最大容量。有两种方式：
① 文件增长不受限制：数据文件可以无限制增大 ② 将文件增长限制为：将文件的大小限制在某一范围内。

1.2.3 事务日志文件的属性

同数据文件。

1.2.4 使用对象资源管理器创建数据库、设置数据库选项

1 、设置数据库名称

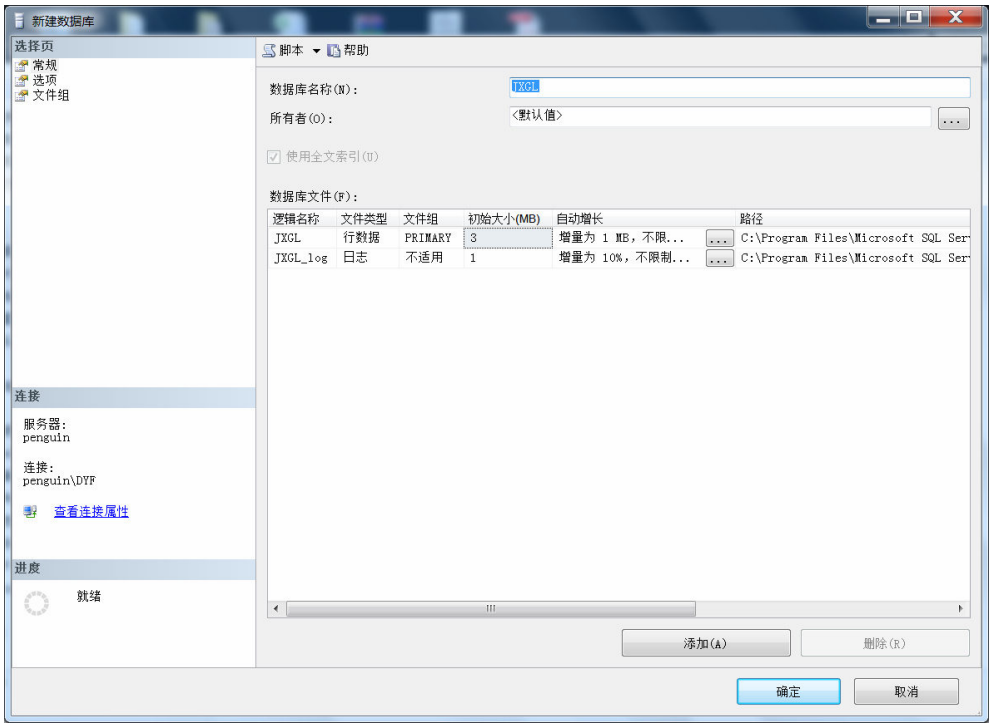


图 1-1 创建数据库之设置常规属性

2 、设置“数据文件”属性

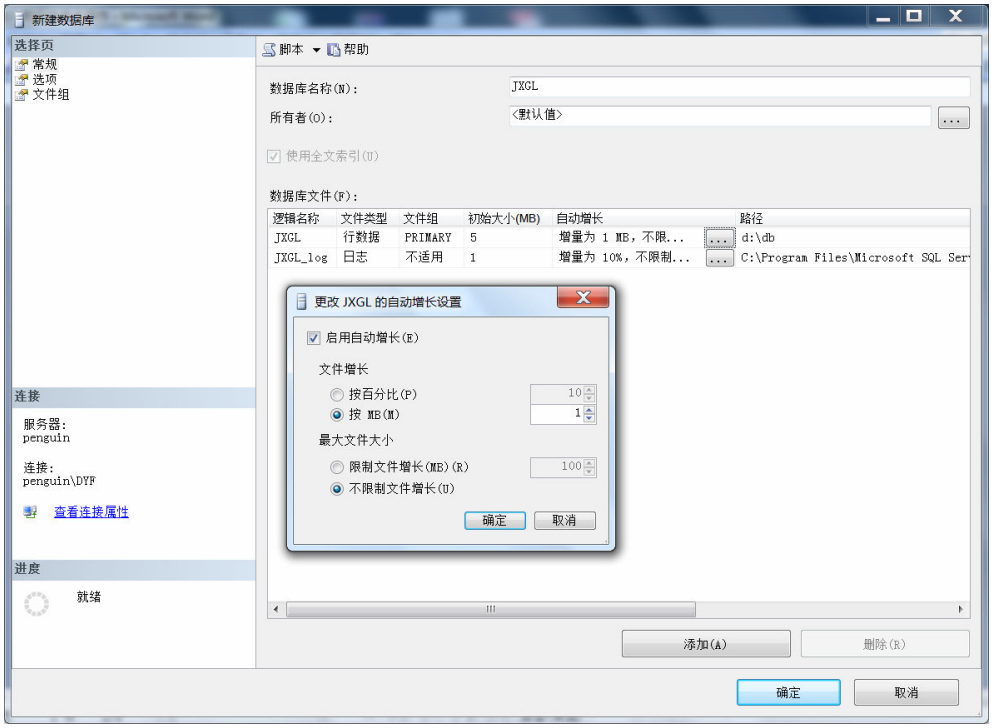


图 1-2 创建数据库之设置“数据文件”属性

3 、设置“事务日志”属性

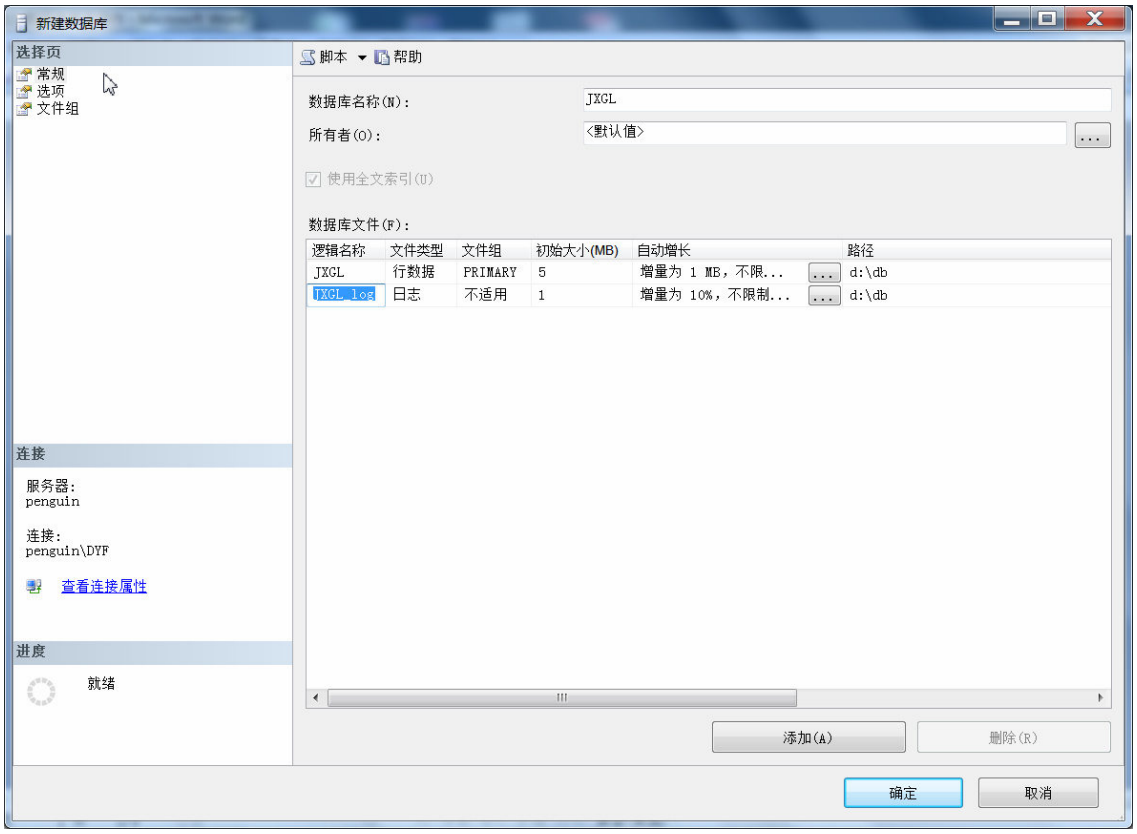


图 1-3 创建数据库之设置“事务日志”属性

1.2.5 使用 T-SQL 语句创建数据库

1 、创建数据库最简单的形式： Create Database 数据库名称

2 、创建数据库完整的语法：

CREATE DATABASE datebase_name 定义数据库的名称
[ON 指定数据库所需的数据文件
[PRIMARY] 其后定义的第一个文件是主数据文件
[<filespec> [,...n]] 定义数据文件
[,<filegroup>[,...n]] 定义数据文件组
]
[LOG ON 指定数据库所需的事务日志文件
{ [<filespec> [,...n]}] 定义事务日志文件
[FOR LOAD |FOR ATTACH]
<filespec> : : = 定义文件格式
([NAME=logical_file_name,] 定义文件的逻辑文件名 ，只在 SQL 语句中使用
[FILENAME='os_file_name',] 定义文件在磁盘中的实际名称和存放路径

[SIZE=size,] 定义文件的初始容量

[MAXSIZE={max_size|UNLIMITED},] 定义文件可以增长的最大容量

[FILEGROWTH=growth_increment])[,...n] 定义文件每次可以增长的容量

<filegroup> : : =

FILEGROUP filegroup_name<filespec>[,...n] 指定文件组及文件组的名称

1.2.6 在对象资源管理器中修改数据库

在对象资源管理器中修改数据库结构，右键点击需要修改的数据库名称打开该数据库的属性对话框，在对话框中修改相应的数据文件属性值。

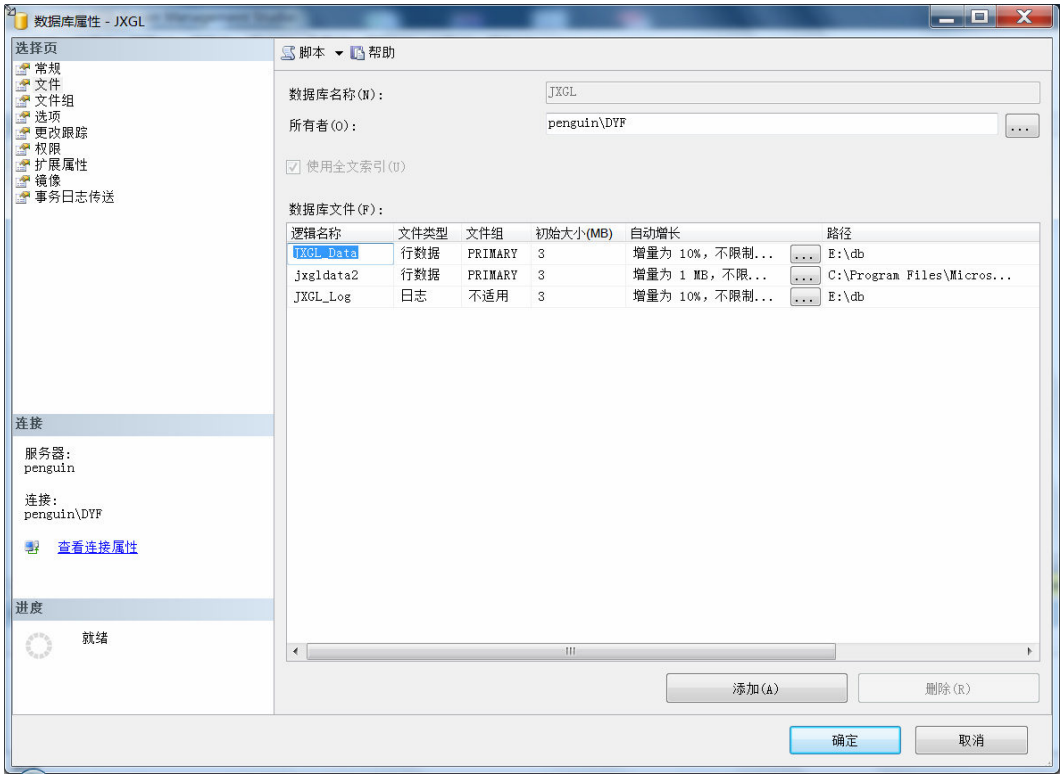


图 1-4 修改数据库之数据文件属性

1.2.7 使用 T-SQL 语句修改数据库

1 、更改数据库名称

sp_renamedb old_name , new_name

说明： @old_name ： 原数据库名称； @new_name ： 新的数据库名称

注意:

① 一般情况下， **SQL SERVER** 是多用户模式。在给数据库更名之前，必须将数据库切换到单用户模式下，更名之后再恢复为多用户模式。

例： sp_dboption 'JXGL','single user','true' ------- 切换成单用户模式

sp_dboption 'JXGL','single user','false' ------- 切换成多用户模式

② 只能用这个方法更改数据库的名字。

2 、缩小数据库文件

DBCC SHRINKFILE

(filename {[,target_size] | [,{|NOTRUNCATE|TRUNCATEONLY}]})

说明:

filename ： 缩小文件的逻辑名称。（可以是数据文件也可以是事务日志文件）

target_size ： 缩小后文件大小。

3 、使用 ALTER DATABASE 语句修改数据库结构

ALTER DATABASE databasename

{ ADD FILE <filespec>[,…n] 增加新的数据文件

[TO FILEGROUP filegroup_name] 将数据文件添加至文件组（该文件组必须已经存在）

|ADD LOG FILE <filespec>[,…n] 增加新的事物日志文件

|REMOVE FILE logical_file_name 删除数据文件或者事物日志文件

|ADD FILEGROUP filegroup_name 增加新的文件组

|REMOVE FILEGROUP filegroup_name 删除已有的文件组

|MODIFY FILE <filespec> 更改数据文件或者事物日志文件的结构（包括扩大数据库容量）

|MODIFY FILE {NAME=file_name,NEWNAME=newname} 更改数据文件或者事物日志文件的逻辑文件名

|MODIFY FILEGROUP filegroup_name {filegroup_property|NAME = new_filegroup_name}} 更改文件组的属性或者文件组名称

<filespec> : : =

([NAME=logical_file_name,][FILENAME='os_file_name',] **该项不能改**

[SIZE=size,][MAXSIZE={max_size|UNLIMITED},][FILEGROWTH=growth_increment])[,...n]

说明:

① 如果要在新增加的文件组内增加数据文件，必须先使用 ALTER DATABASE 语句增加文件组，再使用 ALTER DATABASE 语句将新定义的数据文件添加到该文件组。

② 删除文件组也会同时删除文件组内的数据文件

③ filegroup_property 表示文件组属性

- READONLY ： 指定文件为只读，不允许更新其中的对象，主文件组不能设置为只读。
- READWRITE ： 逆转 READONLY 属性，允许更新其中的对象。
- DEFAULT ： 将文件组设置为默认数据库文件组。

④ 使用 ALTER DATABASE 语句修改数据库结构，每个 ALTER DATABASE 语句只能完成一种操作。(如:ADD FILE 、ADD LOG FILE 等)

1 .2.8 删除数据库

1 、在对象资源管理器中删除数据库

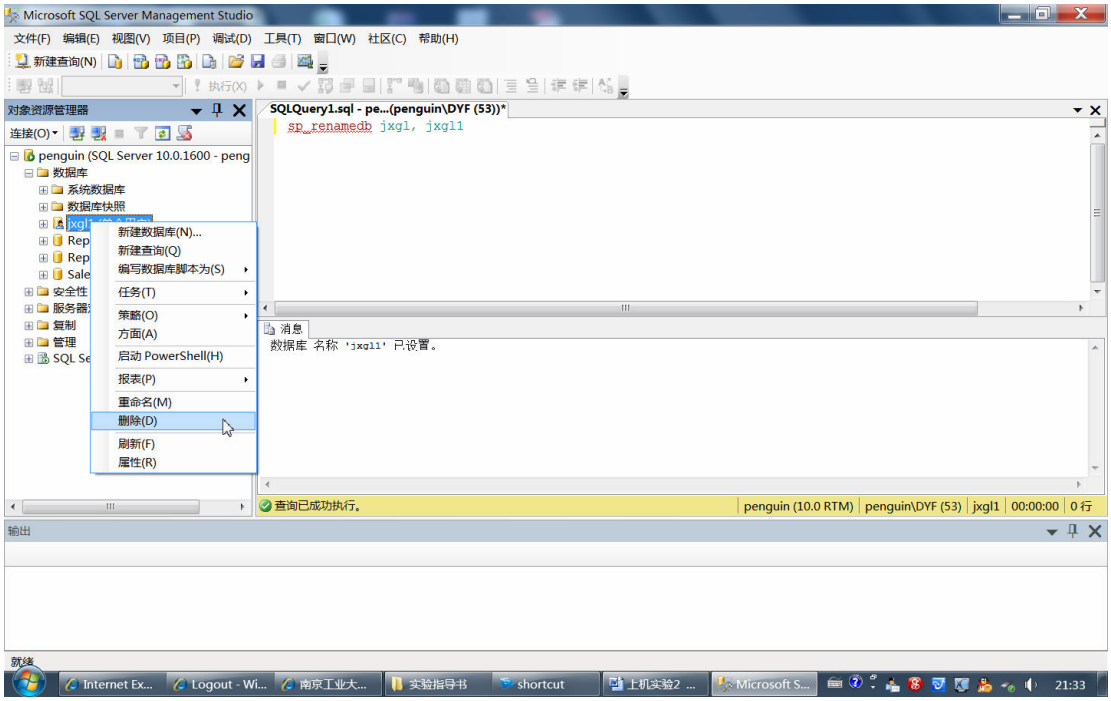


图 1-5 删除数据库

2 、使用 T-SQL 语句删除数据库

DROP DATABASE database_name [,…n] 删除数据库的名称

1.3 实验内容

1.3.1 使用对象资源管理器创建数据库

操作步骤:

(1) 打开对象资源管理器。

(2) 在对象资源管理器控制面板目录中选择“数据库”节点。

(3) 在“数据库”节点上单击右键，并在弹出菜单中选择“新建数据库”， 数据库命名为 JXGL。

(4) 输入目标数据库名称，可进行任意修改。

(5) 在对话框下方数据文件栏中添加一个数据文件:

文件名： test_Data ，位置：默认，初始大小： 5MB， 最大为 50Mb，增长方式为 10%，文件组： PRIMARY。

(6) 在“事务文件”标签添加一个事务日志文件:

文件名： test_Log ， 位置：默认，初始大小：1MB，最大为 10Mb，增长方式为每次增加 1MB。

(7) 关闭数据库属性对话框.

1.3.2 使用 T-SQL 语句创建数据库

创建一个名为 Student 的数据库，定义一个主文件、两个二级文件和两个日志文件。其中主数据文件的逻辑文件名为 stu1，磁盘文件名为 stu1_dat.mdf 。一个二级文件的逻辑文件名为 Stu2 ，磁盘文件名为 stu2_dat.ndf ；另一个二级文件的逻辑文件名为 stu3 ，磁盘文件名为 stu3_dat.ndf 。一个事务日志文件的逻辑文件名为 stu1_log，磁盘文件名为 stu1_log.ldf ；另一个事务日志文件的逻辑文件名为 stu2_log ，磁盘文件名为 stu2_log.ldf 。所有文件的初始容量都是 5MB ，最大容量限制都来是 10MB ，在最大容量限制内，当文件空间不够时每次的增长量都是 1MB 。

操作步骤：

1、 在 SQL Server 主界面中点击“新建查询”启动命令编辑窗口。

2、 在 SQL 命令编辑窗口录入语句（参考预备知识 ）

注意： FILENAME 的值 D: \db \ ……

3、执行 SQL 命令，注意查看“ D: \ db ”目录，生成哪些文件？写出这些文件名，说明哪些是数据文件，哪些是事务日志文件？

4、保存该 T-SQL 命令，文件名 **student.sql** ，存放在 **D: \db。**

1 .3.3 使用 T-SQL 语句设置数据库选项

1、将 student 数据库的选项设置为“ AUTO CLOSE ”

步骤：

(1) 打开命令编辑窗口。

(2) 在 SQL 命令编辑窗口录入语句。

具体语句：sp_dboption 'JXGL','auto close','true'

(3) 执行 SQL 命令，查看结果。

1 .3.4 使用 T-SQL 语句修改数据库

1、在 SQL 命令编辑窗口中创建一个名字为 Temp 的数据库，此数据库包含一个数据文件和一个日志文件，其中数据文件的逻辑名为 Temp1_dat ，磁盘文件名为 Temp1_dat.mdf ，事务日志文件的逻辑名为 Temp1_log ，实际文件名 Temp1_log.ldf ，初始大小为 5MB ，增长上限为 15MB ，每次增长量为 1MB 。（ 提示： FILENAME 的值 D:\db\ ……）

2 、为刚刚创建的名为 Temp 的数据库增加一个数据文件，数据文件的逻辑名称为 Temp2_dat ，磁盘文件名 Temp2_dat.ndf ，数据文件的初始大小是 2MB ，最大增长上限是 12MB ，每次增长量为 2MB 。（提示： ALTER DATABASE Temp ADD FILE ）

3 、为刚刚创建的名为 Temp 的数据库增加一个日志文件，日志文件的逻辑文件名 Temp2_log ，磁盘文件名 Temp2_log.ldf ，文件的初始大小是 2MB ，

最大增长上限是 12MB ，每次增长量为 2MB 。(提示： ALTER DATABASE Temp ADD LOG FILE ）

4 、将修改后的 Temp 数据库中的数据文件 Temp1_dat 的容量增加到 10MB ，并将其容量长上限增加到 12MB ，递增量增加到 2MB 。（提示: ALTER DATABASE Temp MODIFY FILE ）

5 、删除 Temp 数据库中一个名为 Temp2_dat 数据文件和一个名为 Temp2_log 的事务日志文件（提示： ALTER DATABASETemp REMOVE FILE ）

6 、为 Temp 数据库增加一个名为 Temp_ Filegroup 的 文件组，向 Temp 添加一个数据文件 Temp4_dat ，初始容量为 3MB ，最大容量为 10MB ，递增量 1MB ，并且把着这个数据文件添加到 Temp_ Filegroup 文件组中然后再把这个文件组设置为默认文件组。提示:

```
ALTER DATABASE Temp
   ADD FILEGROUP
ALTER DATABASE Temp
   ADD FILE
ALTER DATABASE Temp
   MODIFY FILEGROUP
```

7 、将 Temp 数据库中的数据文件 Temp1_dat 缩小至 5MB（提示：DBCC SHRINKFILE ）。

1.3.5 在对象资源管理器中修改数据库

1、打开 SQL server 主界面，在对象资源管理器中找到数据库“ JXGL”，右键选择“属性”，打开数据库属性对话框;

2、修改 **数据文件和事务日志文件** ，使文件初始化大小为 10MB ，文件的递增量为 5MB ，文件的最大值为 50MB 。

3、 将 JXGL 数据库缩小为原来的 50%
步骤：SQL server 主界面，在对象资源管理器中选定要缩小的数据库，点右键选择 /“所有任务”/“收缩数据库”收缩后文件中的最大可用空间： 50%。

上机实验 2 数据表和表数据基本操作

2.1 实验目的

1、掌握在对象资源管理器中创建表、修改表、操作表中的数据
2、掌握使用 T-SQL 命令创建表、修改表、操作表中的数据
3、通过练习了解并掌握各种常用数据类型的用法

2.2 实验练习预备知识点

2.2.1 数据类型

整型数据类型： bigint 、 int 、 smallint 、 tinyint （明确取值范围）
浮点型数据类型：近似数值型： real 、 float ；精确数值型： decimal 、 numeric （明确取值范围以及表示方法）
字符型数据类型： char 、 nchar 、 varchar 、 nvarchar （明确取值范围， 4 种类型之间的区别）
货币型数据类型： money 、 smallmoney （明确取值范围）
日期和时间型数据类型： datetime 和 smalldatetime （明确取值范围）
文本和图形数据类型： text 、 ntext 和 image （明确作用）
逻辑型数据类型： bit （明确作用）

2.2.2 数据表的基本概念

表是一种数据库对象，用来存储数据。也是数据库中唯一可以存储数据的容器。由 4 个基本部分组成。分别为表名、字段名称（列名）、字段的数据类型、相应的约束。
在 SQL Server 中创建表的方法由两种：
1、利用对象资源管理器来创建；
2、使用 T-SQL 语句创建。

2.2.3 在 SQL Server 企业管理器中创建表

在对象资源管理器中找到对应数据库，选择表对象点右键新建表打开对话框如图 2-1 所示：

1、表字段的一般属性：

① 列名：表中列的名称
② 数据类型：该列对应的数据类型
③ 长度：字段的长度，字段所占的字节数（有些数据类型的长度是固定的如 int）
④ 允许空：表中每条记录该列是否允许为空。

2、表字段的特殊属性（对其中一部分作说明）

① 描述： 指定字段的注释文本描述
② 默认值： 指定字段的默认值，插入记录时没有指定字段值的情况下，自动使用的值

③ **精度：** 指定字段的位数。对于 decimal 和 numeric 数据类型字段可以设置精度属性

④ **小数据位数：** 显示该列值小数点右边能出现的最多数字个数精度和小数数据位数一起使用，来表示 decimal 和 numeric 数据类型数据的取值。

⑤ **标识：** 指定一个字段是否为标识字段。标识字段的数据是由系统自动生成有序数字，可作为表的唯一标识。注意：只能 bigintint、smallint、 tinyint、decimal、numeric 可以设置该属性。

否： 不设置该字段为标识字段。

是： 该字段为标识列，在插入一个新的数据行时不必为字段指定数值，系统会自动生成一个字段值。

是（不适用于是复制）： 和第二个选项功能相似，如果是以复制的方式向表中输入数据，系统将不自动生成字段值。

⑥ **标识种子：** 指定标识字段的初始值

⑦ **标识递增量：** 指定标识字段的递增值。默认值为 1 标识、标识种子、标识递增量需要一起使用来表示字段的标识情况。

⑧ **公式：** 指定用于计算字段的公式

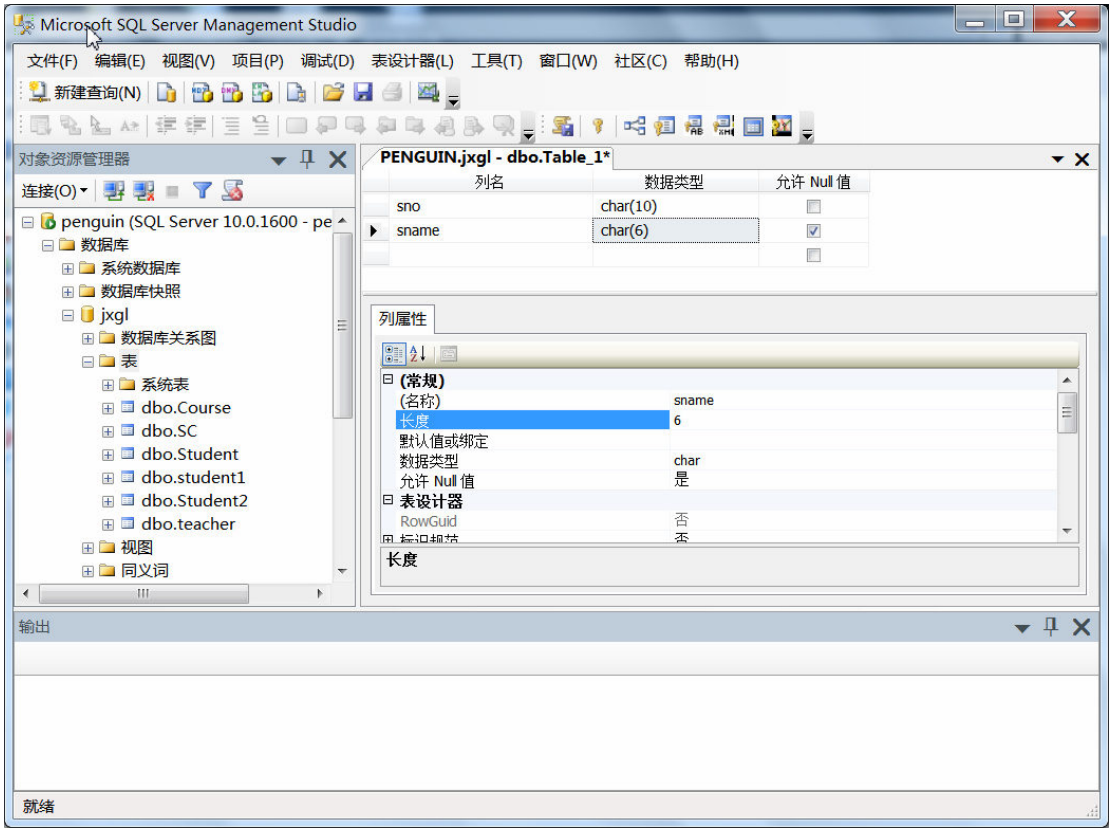


图 2-1 设计表的对话框

按“保存”按钮，输入数据表名称，数据表设计完成。

选中对象表后，点右键选择“编辑前 200 行”，就可以打开表，输入数据。

如图所示 2-2。

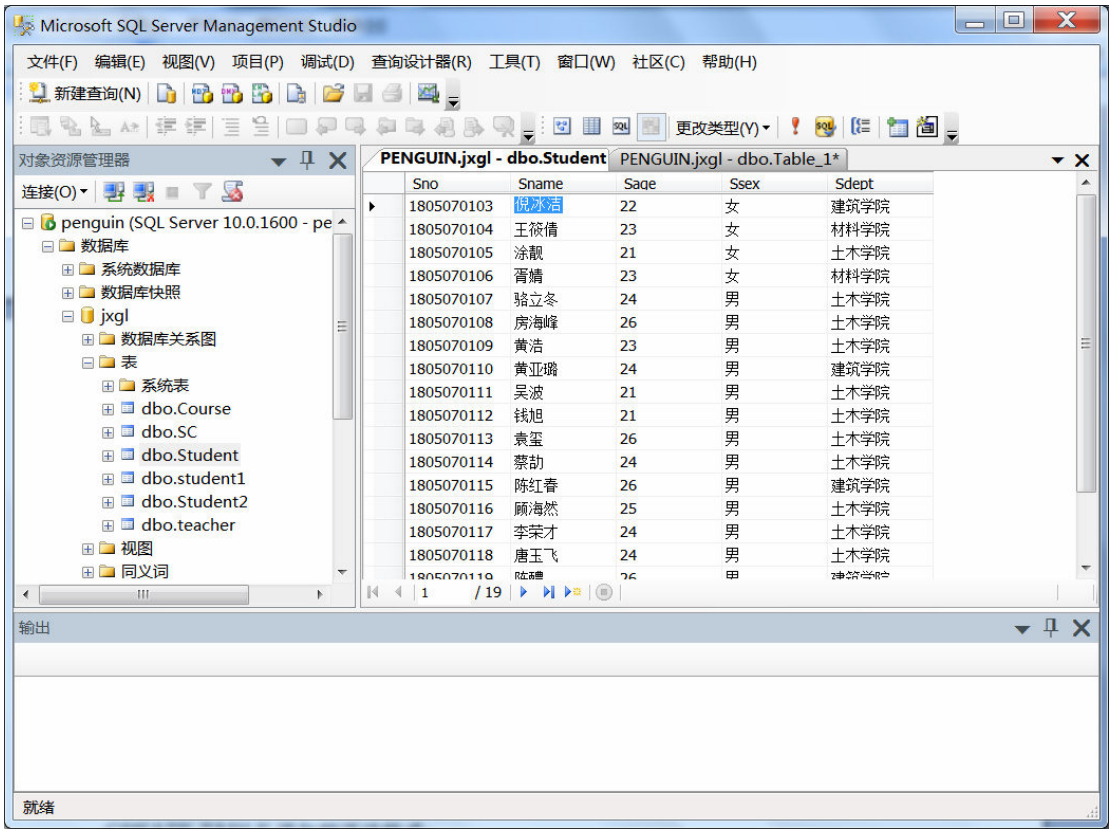


图 2－2 打开的数据表

2.2.4 使用 T-SQL 的 CREATE TABLE 语句创建表

CREATE TABLE 语句的语法格式：

CREATE TABLE [database_name .[owner].| owner .] table_name

表的名称

({<column_definition> 列定义

| column_name AS computed_column_expression 定义计算字段表达式，

|<table_constraint>} 字段约束

[, …n]) [ON { *filegroup* | DEFAULT }] 指定存储表的文件组

[TEXTIMAGE_ON]{filegroup|DEFAULT} 表中 TEXT、IMAGE 数据类型存储的文件组

<column_definition>::={column_name data_type} 字段名称（列名） 对应的数据类型

[[DEFAULT constant_ expression] 设置默认值

|[IDENTITY[(seed,increament)]]] 设置标识

<column_constraint>][…n] 字段约束（下一章详细介绍）

2.2.5 在 SQL Server 对象资源管理器中修改表

在对象资源管理器中打开表的设计试图，直接修改，最后保存。

注意 ：表中有记录后，不要轻易修改表的结构，特别是修改列的数据类型。否则容易引起数据转化出错。

2.2.6 使用 T-SQL 的 ALTER TABLE 语句修改表

ALTER TABLE 的语法格式如下：

```
ALTER TABLE  [ database_name .[ owner ].| owner .] table_name
    ADD {[ column_name data_type [<column_constraint>] ] [,…n] }    在表中增加字段
   |ALTER COLUMN   column_name new_data_type      修改表中字段的数据类型
   |DROP COLUMN column_name [,….n]                  删除表中的字段
```

注意：

① 使用 ALTER TABLE 语句修改数据表时，每个 ALTER TABLE 语句只能执行一种操作（如 ADD 或者 ALTER COLUMN 等）。

② 使用 ALTER TABLE 语句修改数据表时，如果是增加字段的操作，那么字段的定义和创建表时的定义是一样的。

2.2.7 在 SQL Server 对象资源管理器中删除表

在对象资源管理器中打开数据库，选中对应的表通过右键菜单后直接删除。

注意：

①删除表时，表的结构定义、表中所有的数据以及表的索引、触发器、约束等均被永久地从数据库中删除。

②如果要删除有过外键约束和唯一约束和主键约束相关的表，必须首先删除具有外键约束的表。

2.2.8 使用 T-SQL 的 DROP TABLE 语句删除表

DROP TABLE 语句的语法格式如下：

```
DROP TABLE table_name[,…n]
```

2.2.9 使用 T-SQL 语句实现表的数据操作

1、向表中插入（添加）数据

INSERT 语句格式如下：

```
INSERT [INTO] {table_name|view_name} [column_list]
{VALUES({DEFAULT|NULL|expression}[,…n])}
```

说明：

① [INTO]：可选的关键字，用在 INSERT 和目标表之间；

② table_name：需要插入数据的目标表的名称；

③ view_name：视图的名称；

④ column_list：要在其中插入数据的一列或多列的名称列表，列名必须和引用表相对应。列表顺序与 VALUES 列表顺序相吻合。省略该项，则 VALUES 子句给出每一列的值；

⑤ VALUES：为 column_list 列表中的各列指定值。其值有三种： DEFAULT（默认值）、NULL（空值）、expression（常量、变量和表达式）

2、修改表中数据

UPDATE 语句格式如下：

```
UPDATE {table_name|view_name}
    SET  {column_name={expression|DEFAULT|NULL}
    {FROM <table_source>}
    {WHERE<search_condition>}
```

说明：

① table_name：需要插入数据的目标表的名称；

② view_name：视图的名称；

③ SET：指定要修改的列或变量名称的列表。

④ column_name：要更改数据的列的名称；

⑤ expression：变量、常量、表达式或者返回单个值的 SELECT 语句；

⑥ DEFAULTWHERE：使用对列定义的默认值。如果该列没有默认值并定义允许为空，则该列更改为空；

⑦ WHERE <search_condition>：指明只对满足该条件的进行修改，若省略该子句，则对表中的所有进行修改。

3、删除表中数据

DELETE 语句格式如下：

```
DELETE  [FROM] {table_name|view_name}
   {WHERE<search_condition>}
```

说明：

① WHERE<search_condition>：指明只对满足该条件的进行修改，若省略该子句，则对表中的所有进行修改。

②删除表中所有记录和删除表是由区别的。删除表中所有记录后表为空，但是表结构还是存在，但是删除表，不但没有记录，而且表结构也删除了。

2.2.10 在对象资源管理器中实现表的数据操作

在对象资源管理器中打开表后，可进行插入、更改、删除各项操作。

2.3 实验内容

2.3.1 创建数据库 JXGL，方法任意，请参见实验 1（注意：为了后续实验要求，请同学们每次实验课结束时即使将 JXGL 数据库备份）。

2.3.2 使用对象资源管理器在 JXGL 数据库中在创建三张数据表 student（学生表），Course（课程表），SC（选课表）， 表名分别是 student，course，sc，

表结构分别为：

student 表结构：

列名	数据类型及长度	允许为空值	要求	说明
Sno	char(10)	否	设置为主键	学号
Sname	char(6)	否		姓名
Sage	int	是		年龄
Ssex	char(2)	是	设置默认值“男”	性别
Sdept	char(20)	是		学院

Course 表结构：

列名	数据类型及长度	允许为空值	要求	说明
Cno	char(2)	否	设置为主键	课号
Cname	char(20)	否		课名
Ccredit	tinyint	是		学分
Cpno	char(2)	是		先修课号

SC 表结构：

列名	数据类型及长度	允许为空值	要求	说明
Sno	char(10)	否	设置为主键	学号
Cno	char(2)	否	设置为主键	课号
Grade	numeric（5,1）	是		成绩

1、增加一个“奖学金”字段，数据类型为 money ，允许为空。

2、再将“奖学金” 字段删除。

3、向刚才所建的三张表中任意添加若干条记录数据。

注意：为了后续实验要求，三张表中需要相应的数据记录，在输入数据记录时，注意 SC 表中 SNO 字段的值必须和 Student 表中的 SNO 字段取值相对应，也就是说 SC 表中 SNO 字段的值要在 Student 表中的 SNO 字段值中存在；同样 SC 表中 CNO 字段的值必须和 Course 表中的 CNO 字段取值相对应。

4、练习在该数据表中修改、删除记录。

2.3.3 使用 T-SQL 语句管理数据表

1、练习使用 T-SQL 语句在 JXGL 数据库中创建数据表 Stu ,表结构同上 student 表（注意：为了与 student 表区分，这里表名用的是 stu）。

提示： CREATE TABLE ……

2、修改 Stu 表，将 “sname”字段的数据类型更改为 varchar(10) 。

提示： ALTER TABLE …… ALTER COLUMN ……

3、 向 Stu 表添加一个字段，字段名是 “photo”，数据类型是 image，不需要指定字段长度。

提示： ALTER TABLE …… ADD ……

4、删除 Stu 表中的“photo”字段。

提示： ALTER TABLE …… DROP COLUMN ……

2.3.4 使用 T-SQL 语句管理表数据

1 、向 Stu 表 （ sno ,sname, sage, ssex, sdept ）添加一条记录

（ 0405102, 王秋萍, 20, 女,化工 ），注意字符型字段赋值时要用引号。

提示： INSERT [INTO] {table_name }
[column_list] {VALUES({DEFAULT | NULL | expression}[,…n])}

2、再向 Stu 表 添加一条记录，记录的值可以任意取。

3、将 Stu 表 中所有学生的学院调整为“ 测绘 ”。

提示： UPDATE …… SET ……

4、将 Stu 表 中“王秋萍”的性别调整为“男”。

提示： UPDATE …… SET …… WHERE……

5、在 Stu 表 中“王秋萍”年龄在原来的基础上增加一岁 。

6、删除 Stu 表中姓名是 “ 王秋萍 ” 的学生记录。

提示： DELETE[FROM] …… {WHERE<search_condition>}

7、删除 Stu 表中的所有记录。

上机实验 3　结构化查询语句练习

3.1 实验目的

1、练习利用简单的 SELECT…FROM…WHERE 语句查询数据库中的数据
2、掌握 SELECT 语句中关键词的作用和使用方法
3、掌握 WHERE 语句在各种情况下的写法
4、掌握 SELECT 语句的统计函数的作用和使用方法
5、通过练习 SELECT 语句的 GROUP BY 和 ORDER BY 字句的用法，理解其作用，掌握语句的写法
6、通过练习涉及多张表的连接查询，掌握它的作用和写法
7、练习嵌套查询的使用方法，掌握它的特点、语句含义和写法

3.2 实验练习预备知识点

3.2.1　SQL 概述

SQL (Structured Query Language) 的全称是结构化查询语言。
它的功能有：数据定义、数据操纵、数据控制。

3.2.2　SELECT 语句结构

SELECT 语句是 SQL 语言中最基本的查询语句，它主要用来从数据库中检索出符合条件的记录。
SELECT 语句的基本结构如下：

SELECT select_list	要检索的字段列表
FROM table_list	指定检索数据的数据源列表
[WHERE search_coditons]	指定筛选条件，满足该条件的记录才能被检索出来
[GROUP BY group_by_list]	根据 group_by_list 提供的字段的值将结果集分组
[HAVING search_conditions]	定义筛选条件，对分组后的结果集进行筛选
[ORDER BY rdre_list[ASC\DESC]]	定义最终结果集中的记录排列顺序

补充说明：
① select_list ：不但可以包含数据源表或视图中的字段名称，还可以包含其他表达式，如常量或者函数。
② WHERE search_coditons ： WHERE 子句的筛选条件可以是一个或者多个。

3.2.3 SELECT 子句

SELECT 子句是核心语句，用于指定要选择字段的列表。
① 字段列表，字段名称之间用逗号间隔开；
例 1 显示 Student 表中学生的学号，姓名，年龄
SELECT sno, sname, sage FROM Students
② 用 *代替某张表中所有字段
例 2 查看 Student 表中所有的内容
SELECT * FROM Student

③ 包含算术表达式
例 3 检索给 SC（选课表）中所有学生的各课成绩增加 10 分后的结果
SELECT sno, cno, grade = grade+10 FROM SC

④ DISTINCT 关键字。作用是主要用来从 SELECT 语句的结果集中去掉重复的记录。
例 4 检索 Student 表中的班级，去掉重复的记录
SELECT DISTINCT Sdept FROM Students

⑤ TOP 关键字。 主要用来限制返回到结果集中的记录的数目。分为：TOP n（前 n 条记录）和 TOP n percent（前 n%的记录）
例 5 检索 Students 表中前两条记录的学号和姓名。
SELECT TOP 2 sno,sname FROM Student

3.2.4 FROM 子句

FROM 子句主要用来指定检索数据的来源。即指定数据来源的数据表和视图。
注意点：
① 数据库系统中可能存在对象名重复的情况，可使用用户 ID 来限定数据表名称。
例 6 SELECT * FROM dbo.Student
② 在使用 select 语句进行查询时，也可以引用其他数据库中的表，可使用数据库名和用户 ID 来限定数据表名称。
例 7 SELECT * FROM jxgl.dbo.Student

3.2.5 WHERE 子句

WHERE 子句：来指定查询条件，控制结果集记录的构成。

1、WHERE 子句中可以指定运算符号

等于： = ,大于： >,小于：<,不等于：<> 大于等于：>=,小于等于：< =

例 8 检索年龄大于 20 的学生信息

SELECT * FROM Student WHERE sage>20

注意： text, ntext,和 image 数据类型不能与比较运算符组合成查询条件。

2、逻辑运算符 and ,or ,not

例 9 检索年龄为 20 岁或 21 岁学生的基本信息。

分析: 检索记录的限定条件比较多，要注意彼此之间的关系。

SELECT * FROM Student WHERE sage =20 or sage =21

3、范围运算符 BETWEEN…AND；NOT BETWEEN…AND

BETWEEN…AND：要求返回记录的值在这两个指定值的范围内。同时包括这两个指定的值。 NOT BETWEEN…AND：要求返回记录的值不在这两个指定值的范围内。同时不包括这两个指定的值。

例 10 检索 SC 表中成绩在 70-80 之间的学生学号和所选课程。

分析: 检索记录的限定条件在两个值之间，可以用 BETWEEN…AND 表示。

SELECT sno,cno FROM SC WHERE grade BETWEEN 70 AND 80

4、列表运算符：IN/NOT IN 要求查询时返回所有与 IN 以后列表中任意一个值匹配的记录。格式： IN（列表 1，列表 2，….）

例 11 查询‘测绘','建筑'学院的学生学号、姓名。

分析: 设定了 sdept 字段的取值范围，就可以用列表运算符 IN 来表示。

SELECT sno,sname FROM Student WHERE class in('测绘','建筑')

说明： IN 关键字大多数情况下用于嵌套查询，通常首先使用一个 SELECT 语句选定一个范围，然后将选定的范围作为 IN 关键字的符合条件的列表，从而得出最后的结果集。

5、模糊查询条件 LIKE/NOT LIKE

一般用于模糊查询 ,它用来判断某字段值是否与指定的字符串相匹配。使用通配符来表示字符串需要匹配的模式。

例 12 在 Student 表中查询姓为‘王'的学生的学号、姓名

分析: 姓为‘王'的学生，也就是说要求姓名中的第一个字符是“王”，后面跟几个字符并没有做具体的规定。

SELECT sno, sname FROM Students WHERE sname LIKE ‘王%'

说明：通配符

①“%” 匹配任意长度和类型的字符 ②“_” 匹配单个任意字符

③“ []”用于指定范围。④“ [^]”用于指定范围，与“[]”正好相反

6、空值判断符 IS NULL/IS NOT NULL 判断是否为空

例 13 查找成绩表中 Grade 为空的选课记录

分析： TNO 为空也就是 TNO 的值为 NULL。

SELECT * FROM SC WHERE Grade IS NULL

3.2.6 汇总函数

函数语法说明	功能
SUM（[ALL|DISTINCT]表达式）	返回表达式的所有值的和
AVG（[ALL|DISTINCT]表达式）	返回表达式所有值的平均值
COUNT（[ALL|DISTINCT]表达式）	返回表达式中值的个数
MAX（表达式）	表达式的最大值
MIN（表达式）	表达式的最小值

说明：

① 函数中 DISTINCT 的作用是统计计算的过程中去掉重复值。

② 函数中 ALL 的作用是统计计算全部的值包括重复值。可省略。

3.2.7 GROUP BY 子句

作用：将记录根据 GROUP BY 后所跟字段的值分成多个组，进行分组计算。一般情况 GROUP BY 子句与汇总函数连用。

格式： GROUP BY（字段，…n）

例 14 按照年龄把学生信息表的数据分组,并且统计不同岁数的学生人数

分析： Student 表中没有不同年龄人数这一列，因此需要计算才能得到。因此需要先按年龄分组，sage 列中有几个不同的值就要分成几组，再按各组进行统计计算。

SELECT sage,人数= count(*) FROM Student GROUP BY sage

注意：

① 分组也可以根据多个字段；

② 不能对数据类型为 ntext,text,image 或 bit 的字段使用 GROUP BY

1、HAVING 子句

作用： HAVING 子句将对 GROUP BY 子句选择出来的结果进行再次筛选，最后输出符合 HAVING 子句条件的结果。HAVING 子句必须与 GROUP BY 子句连用。

例 15 查询平均成绩在 80 分以上的学生学号

分析： SC 表中没有平均分这一列，因此需要计算才能得到。因此需要先按 sno 分组,sno 列中有几个不同的值就要分成几组,再进行统计计算,最后用 HAVING

子句筛选出 AVG(grade) >80 的记录。

```
SELECT sno, AVG(grade) FROM SC
    GROUP BY sno HAVING AVG(grade)>80
```

2、ALL 关键字

作用：暂时忽略 WHERE 子句中的查询条件。如果使用了 ALL 关键字那么查询结果将包括由 GROUP BY 子句所产生的所有组，无论这些组中的记录是否符合 WHERE 子句中的查循条件。但是对于不符合 WHERE 子句中的查循条件的记录值不进行汇总计算。

例 16 Select sno,平均成绩=AVG(grade) fromSC

```
    where Sno <>'1805060130'
    group by ALL class
```

分析： 使用了 ALL 关键字后，会出现表中所有班级的分组情况，但”1805060130”的平均成绩为空，不再计算 。

说明：正确理解 WHERE 子句、GROUP BY 子句和 HAVING 子句的作用及其作用顺序。

① WHERE 子句用来筛选 FROM 子句中指定的数据源的记录

② GROUP BY 子句将 WHERE 子句的结果集进行分组

③ HAVING 子句将从经过分组后的中间结果集中筛选记录

3.2.8 ORDER BY 子句

作用：在 SELECT 子句用 ORDERBY 子句是对查询结果按照一列或多列进行排序[ASC | DESC]。ASC 表示升序排列，DESC 表示降序排列。默认情况下为升序排列。

例 18 检索 Student 表，并且按照年龄由大到小输出查询结果

分析： 年龄由大到小排列，就表示检索结果要求按年龄作降序排列，那么关键字为 desc。降序排列不是默认的排列方式，需要特别表示。

```
SELECT sno,sname,sage FROM Student ORDER BY sage desc
```

注意：

① Order by 子句中可以指定多个字段，系统将根据子句中排序字段的顺序对查询结果进行嵌套排序。

② Order by 子句中不能包含 text,image 的字段

③ Order by 列表中不允许使用子查询或常量表达式。但可以在选择字段列表中为聚合表达指定一个名称，然后在 Order by 子句引用这个名称。

3.2.9 联合查询（使用 UNION）

作用：联合运算符 UNION 可以将两个或两个以上的查询的结果合并成一个结果集合显示。

语法格式：

```
查询 1……
```

UNION [ALL] ALL：显示所有记录（即使出现重复的记录）
查询 2……

说明：

① 查询的结果的列标题为第一个查询的列标题

② 还可以将不同数据库中表的查询结果联合起来

例 19 如将两个相同表结构的学生表 student 和 stu 记录一起显示出来。

```
select * from student1
union all
select * from stu
```

注意：

① 参加 UNION 运算的结果集必须具有相同的结构，字段数目相同，数据类型兼容

② 使用 UNION 运算符时，单独 SELECT 语句不能包含 Order by 子句。只能在最后一个 Select 语句中使用 Order by 子句，对最终的组合结果集产生作用。

3.2.10 用联接进行多表查询

根据各个数据表之间的逻辑关系从两个或多个数据表中检索数据。联接一种关系运算，它是从两个关系的的笛卡儿积中选取属性间满足一定条件的元组。

说明： 常用的联接表达式为： ([表 1.字段 1]=[表 2.字段 2])

内联接 INNER JOIN

① 利用 JOIN 关键字在 FROM 子句中指定连接条件。

格式：数据表 1 INNER JOIN 数据表 2 ON 联接表达式

例 20 查询学号为‘0311101'的学生的学号，姓名，班级，所选修的课程号和成绩。

分析： 检索的学号，姓名属于 student 表，课程号和成绩属于 sc 表，检索的内容分别属于两张表的情况下，需要先把两张表连接起来，再作检索。首先看这两张表是否可以直接联接，有联接字段？现在 student 表和 sc 表有公共的联接字段 sno，那么就可以实现通过两张表的联接进行多表查询。如果两张没有公共的联接字段，无法联接，那么需要找其他的表，看是否可以通过第三张表将两张表联接起来。

```
Select student.sno , student.sname, sc.cno , sc.grade
    From student inner join sc on student.sno= sc.sno
    Where student.sno=’1805060101’
```

②可以使用 FROM 分句和 WHERE 分句指定数据表之间的连接

```
Select students.sno,sname,cno,grade
    From student,sc
    Where student.sno= sc.sno and student.sno=’1805060101’
```

③为了增加 SELECT 语句的可读性，是 SELECT 语句中的数据表名更加容易识别，可以使用数据表别名

```
Select A.sno, A.sname, A.class, B.cno, B.grade
    From student as A,sc as B
    Where A.sno= B.sno and A.sno=‘1805060101'
```

3.2.11 嵌套查询

一个 SELECT…FROM ..WHERE 称为一个语句块，将一个查询块套在另一个查询块中称为嵌套查询；也就是一个外层查询包含一个内层查询，称外层查询为外查询，内层查询为子查询。

表示：

```
[SELECT…
    FROM… 外层查询
    WHERE….
        (select…
            from…. 内层查询
            where…)]
```

1、带有比较运算符的子查询（又称单一行的子查询）

定义：如果父查询与子查询之间用比较运算符进行联接（ <,>,<>,=）

例 22 在 SC 表中查询成绩大于平均分的所有选课记录。

分析： 题中要求根据与平均分比较的结果显示记录，但是 sc 表中并没有平均分这一列，不能直接比较，所以需要先计算平均分，再将这个计算的结果作为查询条件再次进行检索。

```
SELECT * FROM SC
    WHERE grade >= (select AVG(grade) from SC )
```

注意： 带有 “比较运算符”的子查询都返回单一的数据

2、多行子查询

当子查询返回的是多条数据，即子查询的结果是一个集合，父查询与子查询之间必须使用列表比较运算符。（ IN、NOTIN、ALL、ANY）

① IN、NOTIN

例 23 查询所有选修了 1002 号课程的学生姓名

分析：已知选修课程号检索学生姓名，无法直接在一张表（ sc 表）中直接进行检索。因为它涉及到 student 和 sc，因此需要根据课程号在 sc 表中检索出对应的 sno，再根据 sno 在 student 表中检索出 sname。但是在第一次的检索 sno 的结果中，sno 不是单一的值，是多个值。因此不能直接使用比较运算符，只能使用列表比较运算符 IN。

```
Select sname From student
```

Where sno in (select sno from sc where cno=’1002’)

*② ALL、ANY

- >ANY 大于子查询结果中的某一个值，只要比最小的大就可以了
- <ANY 小于查询结果中的某一个值，只要比最大的小就可以了
- =ANY 等于查询结果中的某一个值（等同于 IN）
- >ALL 比最大值大就可以了
- <ALL 比最小值小就可以了

例 24 查询测绘学院中比建筑学院中某一学生年龄小的学生的基本信息

```
SELECT * FROM Student
    WHERE sdept = '测绘' and
          sage <any(select sage from Student WHERE sdept = '建筑')
```

*3、相关子查询

相关子查询的子查询和父查询之间的连接谓词为 EXISTS 和NOT EXISTS。目的：进行测试，测试子查询是否会返回结果。有结果返回 Ture，无结果返回 False。子查询实际上不产生任何数据。

相关查询的一般处理过程：

首先提取外层查询中表的第一个元组，根据它与外层查询相关的属性处理内层查询，若 WHERE 子句返回为真，则取此元组放入结果集合，然后再取下一个元组。

例 25 查询选修了课程的学生姓名

分析： 学生姓名和选课情况分别在两张表中，查询无法一次完成。可以选择多中方法来完成。如果选择带 EXISTS 关键字的相关子查询，那么先从 Student 选取一条记录，看它是否满足内层查询，如果满足条件即 exists，返回 Ture，输出这条记录的值；如果不满足条件即 not exists，返回 False，那么在 Student 中选取下一条记录，继续判断，直至全部记录取完。

```
SELECT sname FROM Student
    WHERE exists (select * from SC where Student.sno=sc.sno)
```

说明： NOT EXISTS 与 EXISTS 正好相反。

3.2.12 嵌套查询应用于插入数据表中的数据

例 26 将 Stu 表中'测绘'学院的学生信息（sno,sname,sdept）添加到学生表 student 中。

分析： 将一个表中的数据复制到另一个表中，如果一条记录一条记录的复制，效率太低。因此，可以用 select 语句将表中记录筛选出来，再一起插入到另一张数据表中。

```
insert into student(sno,sname,sdept)
    select sno,sname,class from stu where sdept='测绘'
```

3.2.13 嵌套查询应用于更新数据表中的数据

例 26 将选修了 1002 课程且成绩高于课程平均分的学生的基点（假设 sc 表中代表基点的 point 字段）设置为 1。

分析： sc 表中没有每门课程的平均分，因此需要利用子查询先计算出课程平均分 avg(grade)，再根据这个平均分来更新表中的数据。

```
update sc set point=1 where cno=’1002’ and grade>=(select
avg(grade) from sc where cno=’1002’ )
```

3.2.14 嵌套查询应用于删除数据表中的数据

例 27 删除'张三'在 sc 表中的选课记录

分析： 给定的条件是姓名，但是 sc 表中并没有姓名，无法直接修改。因此需要应用子查询先在 Student 中查询出对应的 sno，再根据该值在 sc 中进行删除操作。

```
delete from sc where sno in ( select sno from student where sname='
张三')
```

3.3 实验内容

3.3.1 附加数据库： 将数据库 jxgl 数据文件附加到本地服务器

3.3.2 （简单的结构化查询语言练习）将查询语句写在空白处，在查询分析器上验证。

01 查询所有学生的全部信息（简单查询）

02 查询 Student 表中学生姓名(sname)和性别(ssex)信息

03 显示把 SC 表中各科学生的成绩加 10 分后的信息

04 查询 来自“测绘”学院学生的学号和姓名（条件查询）

05 查询选修“1005”号课程且考试成绩在 70 分以上的学生的学号以及所选修的课程号和成绩

06 查询选修 1003 号课程成绩在 80-90 分之间的学生学号和成绩

07 查询年龄在 18-22 岁之间的学生的详细情况（ 谓词查询 between… and… ）

08 查询学生信息表中姓 ‘张'的学生的基本情况

09 查询学生信息表中姓 ‘张'，姓名列包含两个字的学生的基本情况

10 查询名字中第 2 个字为‘军'的学生的基本信息

11 查询测绘学院和建筑学院学生的姓名和性别（谓词查询 in ）

12 查询既不是测绘学院也不是建筑学院的学生的姓名、性别和所在部门（两种方法）

13 查询学生选课成绩表（ SC ）中缺少成绩的学生的学号和相应的课程号 (is null)

15 查询选修了‘1001 '或者‘ 1002 '课程，成绩及格的学生的学号

3.3.3 （复杂的结构化查询语句练习）将查询语句写在空白处，在查询分析器上验证

001 统计学生信息表中的学生人数（ count() 函数）

002 统计学生信息表中的部门个数（ count() 函数）

003 统计学生信息表中所有学生“年龄”的平均年龄、最大年龄、最小年龄（ avg( ),max( ),min( ) ）

004 统计每个学院的学生人数（ group by 和 count （）函数）

005 分别统计每个学院的男女人数（要求显示出学院、性别、对应的人数）

006 统计学生人数超过 10 人的学院名称和对应的人数（ group by …… having ……和 count 函数）

007 计算选修了“ 1004 ”号课程的学生的平均成绩（要求显示出课程号和对应平均成绩）

008 查询全体学生情况，查询结果按所在学院升序排列，同一学院中的学生按年龄降序排列（order by）

009 统计每位同学的选课门数、平均成绩、总成绩、最高成绩、最低成绩，按照总成绩降序排列

010 查询选修课程学生的学号、姓名、所选课程号、取得的成绩（ inner join ）

011 查询“ 测绘”学院同学的所有信息，如果有选修课的话还要对应的求列出选修的课程号以及成绩。

012 查询选修了‘数据库原理'课且成绩及格的学生的学号、姓名和成绩（复合条件连接）

013 查询每位同学（查询结果含学号和姓名）所学的课程名和对应的成绩，并按照成绩升序排列。

014 查询平均成绩大于 70 分的课程，要求显示课程号和平均成绩。

015 查询选修了“数据库原理”课程且成绩没有及格的学生的学号、姓名以及对应的成绩。

3.3.4 （综合性结构化查询语句练习）将查询语句写在空白处，在查询分析器上验证

001 查询和 “ 张飞 ” 在同一个学院的学生学号和姓名（ 嵌套查询 ）

002 查询“测绘 ”学院中年龄比“苏有朋”大的学生的姓名（ 嵌套查询 ）

003 在 Student 表中查询大于平均年龄的学生学号、姓名、班级、入学成绩（ 嵌套查询 ）

004 查询所有选修了 1002 号课程的学生姓名（ 嵌套查询 in)

005 查询选修了课程名为 “ 数据库原理 ” 的学生学号和姓名（ 多层嵌套 ）

006 在“测绘”学院中查询比“建筑”学院中某一学生“年龄”小的学生的基本信息（ <any ）

007 查询其他学院中比测绘学院所有学生年龄都小的学生姓名、学院和年龄（ >all ）

008 查询所有选修了 1002 号课程的学生姓名（ exists ）

注意比较 008 题和 004 题的查询结果是否相同?

009 查询没有选课的学生的姓名（ not exists ）

010 将选修“1002 ”课程且成绩没有及格的学生的成绩调整为 60 分。

011 删除 ‘测绘' 学院学生在 SC 表中的选课记录

012 查询选修了 3 门以上课程的学生的姓名

013 查询除测绘学院以外的学生的各科课程成绩（学号、学院、课程号、成绩）。

上机实验 4　　视图与索引操作

4.1 实验目的

1、练习并掌握使用对象资源管理器创建和管理视图；

2、练习并掌握使用 Transact_SQL 语言创建和管理视图；

3、练习通过视图操作表中数据，并能理解它的含义；

4、通过在对象资源管理器创建和管理索引，掌握创建和管理索引的多种方法；

5、通过练习使用 Transact_SQL 语言创建和管理索引，理解语句意思，掌握其用法；

6、了解查看索引信息的方法。

4.2 实验练习预备知识点

4.2.1 视图的概念

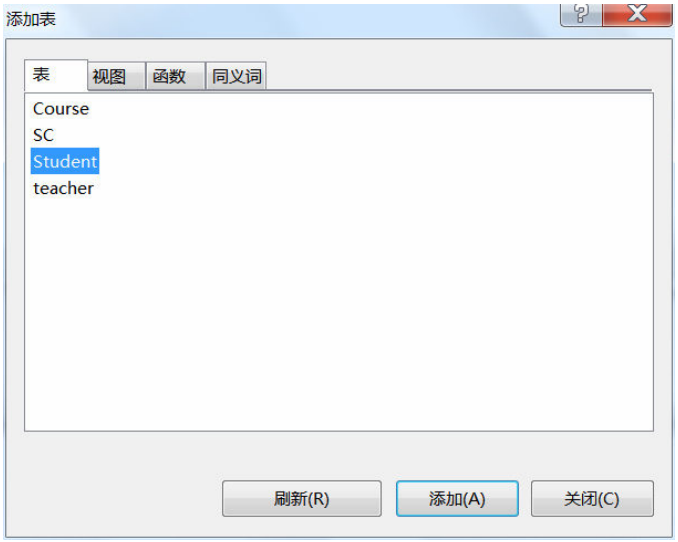
视图是一种数据库对象，是虚拟表。它是从基本表或者视图中导出的表 ，用来存储用户想到看的数据。视图是用户查看数据库的一种方式。

视图与数据表之间的区别：

视图是引用存储在数据库中的查询语句时动态创建的，它本身并不存储数据，真正的数据依然存储在数据表中。

4.2.2 在对象资源管理器中创建视图

1、打开对象资源管理器，在左边的目录树中选择要创建数据库文件夹，选中“视图”对像。点击右键选择“新建视图”，打开“视图”设计窗口。如图 4-1 所示



4-1 视图设计窗口

2、将用来创建视图的表添加到视图设计窗口，如图 4 -2 所示。如果创建的视图覆盖多张数据表，那么这些数据表必须是存在联系的。

3、从数据源中选择将在视图中显示的字段，同时出现在第二部分的列表中。在该列表中可设置限制视图中显示数据的条件，排序字段和排序顺序，以及该字段是否显示。

4、在第三部分的空白处会自动显示出创建该视图所用到的 SELECT 语句。可做修改。

5、在第四部分的列表中，可以预览视图建成后打开的内容。

6、保存视图，取名。

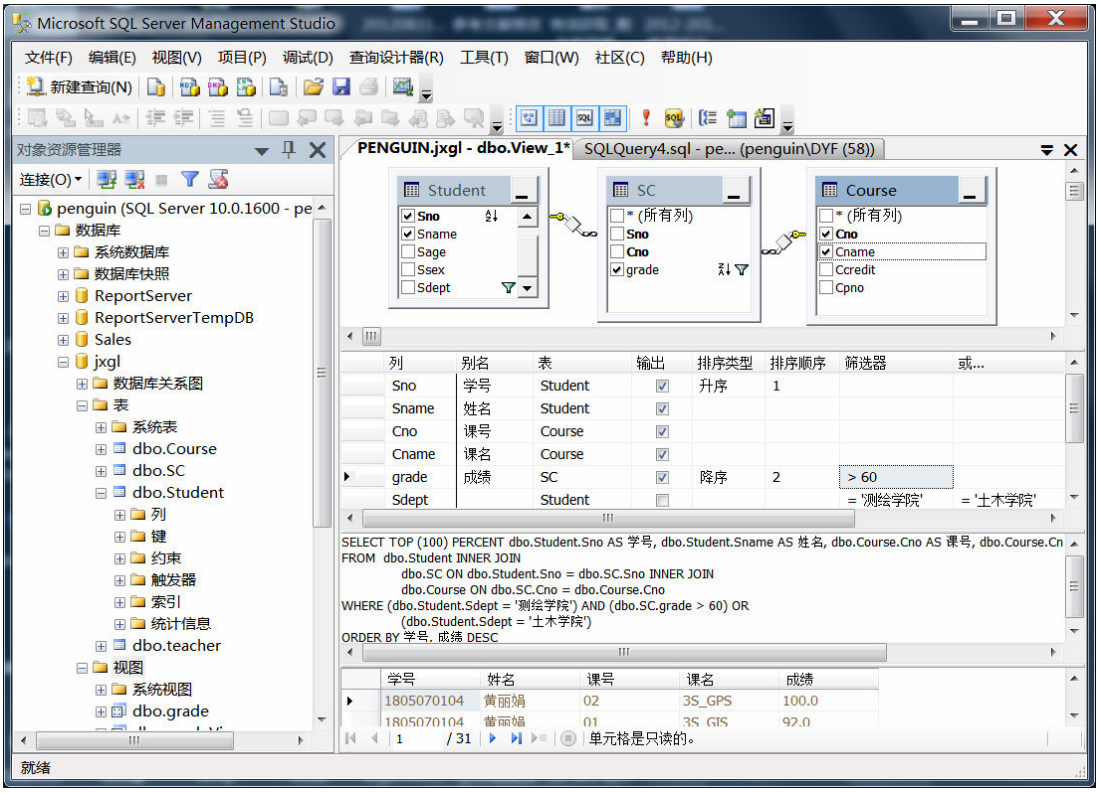


图 4-2 填写好的视图设计窗口

7、最后，打开视图的方法和打开表的方法是一样的。

8、也可以使用向导创建视图（步骤略）

4.2.3 在查询分析器中用 T-SQL 语句创建视图

CREATE VIEW 语句结构如下：

```
CREATE VIEW view_name[(column[,…n])]
    [with ENCRYPTION]
AS
    select_statement
    [WITH CHECK OPTION]
```

参数说明：

view_name：视图名称

column[,…n]：视图中的字段名，可省略。如果没有指定，则视图字段将获

得与 SELECT 语句中的字段相同的名称。

with ENCRYPTION：对创建视图的 T-SQL 语句进行加密。加密过程是不可逆的。

select_statement：定义视图的 SELECT 语句，它可以使用数据库中的不同数据表和其他视图。

WITH CHECK OPTION：强制所有通过视图修改的数据满足 SELECT 语句中指定的选择条件。

注意： ① 下列情况下，有必要指定视图中的字段名：

- 视图字段是算术表达式、函数或常量等计算得到的；
- SELECT 语句返回的结果集中包含两个或更多相同的名称的字段，要给某个字段指定一个不同于基表的字段。

例 1 创建视图，该视图显示测绘学院学生的学号，姓名， 班级信息（并强制检查）

```
CREATE VIEW tmStuView(学号,姓名,班级)
AS SELECT sno,sname,sdept FROM Student
        WHERE sdept='测绘学院'
WITH CHECK OPTION
```

说明：增加了 WITH CHECK OPTION 语句后，通过该视图修改 Student 中信息时，必须满足 sdept='测绘学院'这个条件。

4.2.4 修改视图的定义

1、在对象资源管理器上直接修改

启动对象资源管理器中，打开视图所在的数据库，选择并打开“视图”对象。选择要修改的视图，点右键在选择“设计视图”，打开视图设计器，如图 4－3 所示

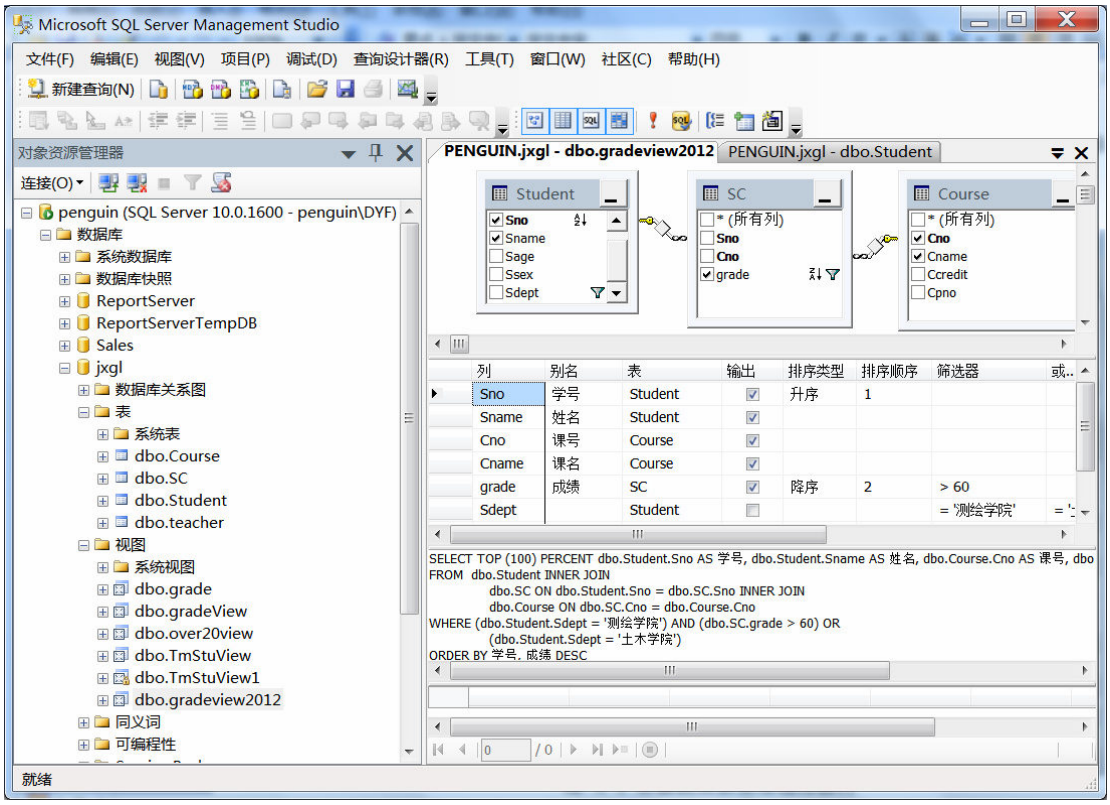


图 4-3 可做修改的视图设计窗口

可直接在该设计视图上修改，保存即可。

2、使用 ALTER VIEW 修改视图定义 ALTER VIEW 语句结构如下：

```
ALTER VIEW view_name[(column[,…n])]
[WITH ENCRYPTION]
AS
    Select_statement
[WITH CHECK OPTION]
```

可以看到，视图修改的语法结构和创建的语法结构几乎相同，用 ALTER 关键字，这里的视图名是已经存在的视图的名。如果是对 Select_statement 做修改，需要重新定义该语句。

4.2.5 删除视图

1、在对象资源管理器中删除视图

启动对象资源管理器中，打开视图所在的数据库，选择并打开“视图”对象。选择要修改的视图，点右键在选择“删除”，打开删除视图窗口，点击“确定”即可。

2、使用 DROP VIEW 删除视图

DROP VIEW 语句结构如下：

DROP VIEW view_name[,…n]

说明：可删除一个或者多个视图。

4.2.6 视图重命名

1、在对象资源管理器中重命名视图

启动对象资源管理器中，打开视图所在的数据库，选择并打开“视图”对象。选择要修改的视图，点右键在选择“重命名”，即可完成。

注意： 视图重命名后，一切引用视图原来名称的语句，都将失去作用。

2、用 T-SQL 语句重命名视图

重命名视图的语句结构如下：

sp_rename[@objname=]'object_name'

[@newname=]'new_name'[,[@objtype=]'object_type']

该语句与数据库重命名的语句相同，参看数据库重命名。

4.2.7 索引概念

索引是对数据库中一个或多个字段的值进行排序的结构。

一个索引就是一个列表，包含值和这些值所在的记录在数据表中的位置。它能够提高检索的速度，改善数据库性能。

1、索引类型：

① 聚集索引（ clustered index）

② 非聚集索引（ nonclustered index ）

2、索引的其他类型

① 唯一索引

② 复合索引

4.2.8 创建索引

1、系统自动创建索引

① 创建或者修改数据表时，当一个字段或者字段组合设置唯一约束，那么系统将自动创建非聚集的唯一索引（ NONCLUSTERED）。

② 创建或者修改数据表时，当一个字段或者字段组合设置主键约束，系统将自动创建聚集的唯一引（ CLUSTERED），前提是表中没有聚集索引。

2、在对象资源管理器中创建索引

启动对象资源管理器中，打开数据表所在的数据库，选择要创建索引的数据表下的“索引”条目，点右键在选择“新建索引”，出现“新建索引”窗口，如图 5－4 所示。

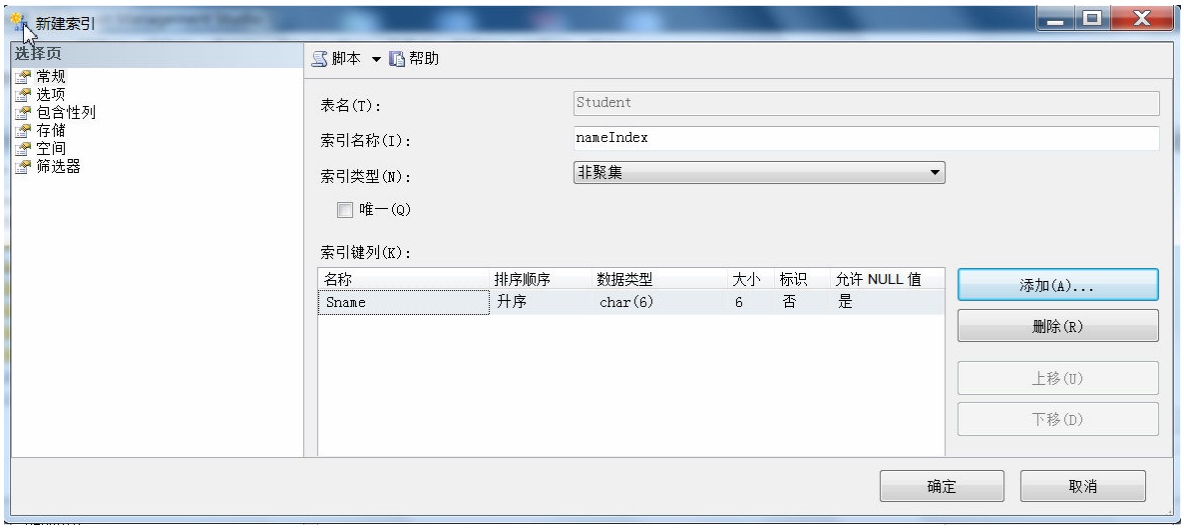


图 4－4 新建索引窗口

在该窗口中可以添加索引，按“添加”按钮，也可以删除索引，按“删除”按钮。在该窗口输入所建索引的名称，选择创建索引的字段。并设置索引的属性。如是否为聚集索引（表中已有聚集索引的话，该选项为灰色），是否为唯一索引，是否为填充索引以及填充因子等。最后选择“确定”即可关闭窗口，并在窗口中看到创建的索引。

3、使用 Create Index 语句创建索引

使用 Create index 创建索引的完整语法：

```
CREATE [Unique][Clustered|nonClustered] INDEX index_name
ON table_name(column[,……])
[ WITH [PAD_INDEX]
    [,FILLFACTOR=fillfactor][,IGNORE_DUP_KEY[,DROP_EXISTING][
,] ]
[ON filegroup]
```

例 6 基于 sc 表中的 sno,cno 字段组合创建唯一索引，忽略重复值。

```
create unique index ix_sc_noname
on sc(sno,cno)
with ignore_dup_key
```

4.2.9 删除索引

1、在对象资源管理器中删除索引

点击目标索引，右击删除。

2、用 DROP INDEX 语句删除索引

DROP INDEX 语句的语法：

DROP INDEX table.indexname [,……]

table.indexname:索引所在的表名和索引名。

例 7 删除上例 6 所创建的索引

Drop index sc.ix_sc_noname

4.3 实验内容

4.3.1 在对象资源管理器中完成下列操作：

1、使用对象资源管理器直接创建视图

在 JXGL 数据库中，基于表“Student”，“Course”，“SC” 创建视图 GradeView2 ，包含字段：**“Sno”，“Sname”，“Cno”，“Cname”，“Grade”**其中在 **“Sdept”** 设置筛选条件为 **“建筑学院”** ，但要求该字段不在视图中显示出来。

2、在对象资源管理器中修改视图

① 修改视图 GradeView2 ，使该视图按照 **“Grade”** 升序排列。

② 重命名视图 GradeView2 为“学生成绩视图”

4.3.2 使用 T-SQL 语句完成下列操作：

1、基于 **student** 表创建视图“学生信息” ，该视图的结果集中包含的字段 **（ sno ， sname ，sage， sdept ）**的值，为该视图设置中文别名，为 **视图文本加密** 。

2、基于 **student,course,sc** 表创建视图：**gradeView3** ，显示的内容包括学生的学号，姓名，课程编号，课程名称，成绩。该视图的结果集中包含的字段 **（student.sno， student.sname， course.cno，course.cname，sc.grade ）** 的值。视图的字段名称用中文表示。

3、将视图 gradeView3 重命名为“学生选课信息” **（ sp_rename ）**

4、调用存储过程查分别看视图 **“学生信息”** 和 **“学生选课信息”** 的视图定义文本。有何差别？写出查看视图定义文本的命令 **（ sp_helptext ）**

5、修改视图 **“学生信息”** ，使之显示 **“ 材料学院 ”** 的学生信息（显示字段不变）， **设置强制检查** 。

6、引用 **“学生选课信息”** 视图，查询选修了 **‘GIS '课程** 的学生姓名和对应成绩。

7、通过 **“学生信息”** 视图向 **student** 表添加一条记录。（记录内容自定义）想一想：添加记录时有哪些限制？

8、通过 **“学生选课信息”** 视图分别向 **student** 表和 **course** 表添加一条记录。（记录内容自定义）想一想：新添加的记录是否会在 **“学生选课信息”** 视图中出现，为什么?

9、通过 **“学生选课信息”** 视图修改 **course** 表中课程号为‘1002’的课程名称（课程号和修改的课程名称可自定义）。

10、通过 **“学生选课信息”** 视图将 **“李四”** 的**“GPS”**课程的成绩修改为 **89**

11、通过 **“学生信息”** 视图将学生姓名为 **“王五”** 的记录删除。

4.3.3 在对象资源管理器中完成下列操作：

在 **对象资源管理器中** 创建索引 IX_N_P

在 **JXGL** 数据库中，为 **SC** 表的“Sno“、“Cno“建立唯一索引。

步骤：选中目标表下的“索引”条目右键点击“新建索引”

5.3.4 使用 T-SQL 语句完成下列操作（管理索引）：

（ 1）在 **JXGL** 数据库中创建表 **Teacher** ，包含字段：Tno，char(4) ，Tname **char(8)** ，Tdept **varchar(20)** 。其中将 Tno 设置为主键；将 Tname 设置为唯一索引。

提示：用 create table 语句并在相应字段后加相应的 Primary Key 或 Unique 关键字。

（ 2）调用存储过程查看 **Teacher** 表的索引情况 **（ sp_helpindex ）** 说明索引的名称和类型。

（ 3） 在 **JXGL** 数据库中为 **course** 表中的 **cname** 字段创建唯一索引，忽略重复值。

提示：用 create index 语句并加相应的关键字，索引名自己定义。

（ 4） 在 **JXGL** 数据库中为 **sc** 表中的 **sno,cno** 字段组合创建唯一索引。

提示：用 create index 语句并加相应的关键字，索引名自己定义。

（ 5）删除第（3）题和第（4）题中创建的索引。

上机实验 5　数据完整性管理

5.1 实验目的

1、理解数据库完整性约束的概念和原理；

2、掌握声明型数据完整性和过程型数据完整性的实现方法；

3、通过练习正确理解触发器的作用，类别，如何产生作用；

4、通过练习熟悉创建触发器的语句。

5.2 实验练习预备知识点

5.2.1 完整性的概念

数据完整性（Data Integrity）是指数据的精确性（Accuracy） 和可靠性（Reliability）。它是应防止数据库中存在不符合语义规定的数据和防止因错误信息的输入输出造成无效操作或错误信息而提出的。

5.2.2 完整性的类型

1 实体完整性（Entity Integrity）

实体完整性规定表的每一行在表中是惟一的实体。表中定义的 UNIQUE PRIMARYKEY 和 IDENTITY 约束就是实体完整性的体现。

2 域完整性（Domain Integrity）

域完整性是指数据库表中的列必须满足某种特定的数据类型或约束。其中约束又包括取值范围、精度等规定。表中的 CHECK、FOREIGN KEY 约束和 DEFAULT、 NOT NULL 定义都属于域完整性的范畴。

3 参照完整性（Referential Integrity）

参照完整性是指两个表的主关键字和外关键字的数据应对应一致。它确保了有主关键字的表中对应其它表的外关键字的行存在，即保证了表之间的数据的一致性，防止了数据丢失或无意义的数据在数据库中扩散。参照完整性是建立在外关键字和主关键字之间或外关键字和惟一性关键字之间的关系上的。在 SQL Server 中，参照完整性作用表现在如下几个方面：

禁止在从表中插入包含主表中不存在的关键字的数据行；

禁止会导致从表中的相应值孤立的主表中的外关键字值改变；

禁止删除在从表中的有对应记录的主表记录。

SQL Server 提供了一些工具来帮助用户实现数据完整性，其中最主要的是：约束（Constraint）、缺省值（Default）、规则（Rule）、和触发器（Trigger）。触发器将在后面的章节中介绍。

5.2.3 约束（constraint）

1 DEFAULT 约束

使用 DEFAULT 约束，如果用户在插入新行是没有显示为列提供数据，系统会将默认支赋给该列。默认值约束的定义格式为：

```
[CONSREAINT constraint_name]
DEFAULT constant_expression
```

其中，constraint_name 参数指出所建立的默认值约束名称。
Constant_expression 表达式为列提供默认值。在使用默认约束是，还应该注意以下两点：1. 每列只能有一个默认约束。2. 约束表达式不能参照表中的其他列和其他表、视图或存储过程。

例 1：新建 teacher 数据表，并在 Tsex 字段上添加缺省约束，其缺省值为‘男’。
create table teacher(Tno char(4), Tname char(6), Tsex char(2) constraint df_sex default ‘男’,Tdept char(20))

例 2：修改 teacher 数据表，并在 Tdept 字段上添加缺省约束，其缺省值为‘测绘系’。
Alter table teacher add constraint DF_dept ‘测绘系’ for Tdept

例 3：修改 teacher 数据表，将缺省约束 DF_sex 删除掉。
Alter table teacher drop constraint DF_sex

2 PRIMARY KEY 约束

在数据库的每个表中，经常有通过一列或者多个列，唯一的标识表中的每一行。就好像我们平时使用的身份证，能够唯一的标识每个人一样。这样的一列或者多个列，被称为主键，通过主键，可以强制表的实体完整性。每一个表中只有一个 PRIMARY KEY 约束，更简单的说，他是通过建立唯一索引保证指定列的实体完整性。在使用 PRIMARY KEY 约束时，该列的空值属性必须定义为 NOT NULL，也就是说拥有主键的那一列，不能为空。由于 PRIMARY KEY 约束确保唯一数据，所以经常用来定义标识列。标识列就是表中已经指派了标识属性的列。标识属性生成唯一数字。

建立主键不仅可以保证表内数据的完整性，而且在为表建立主键的同时，Microsoft SQL Server 能够通过为主键创建唯一索引强制数据的唯一性。如果在 PRIMARY KEY 约束中未指定索引类型时，默认情况下所建立的索引为簇索引。该索引只能通过删除 PRIMARY KEY 约束或其相关表的方法来删除，而不

能使用 DROP INDEX 语句删除。当 PRIMARY KEY 约束由另一张表的 FOREIGN KEY 约束引用时，不能删除 PRIMARY KEY 约束；要删除它，必须先删除 FOREIGN KEY 约束。

通常建立一列约束时，我们称之为列级 PRIMARY KEY 约束，应用于多列时，称之为表级 PRIMARY KEY 约束。列级 PRIMARY KEY 约束的定义格式为：

```
[CONSTRAINT constraint_name]
PRIMARY KEY [CLUSTERED | NONCLUSTERED]
```

表级 PRIMARY KEY 约束定义风格为：

```
[CONSTRAINT constraint_name]
PRIMARY KEY [CLUSTERED | NONCLUSTERED]
{(column[,…n])}
```

例 4：新建 department（部门）数据表，并在 Dno（部门编号）字段上建立聚簇型主键约束 pk_no。

create table department (Dno char(2) constraint pk_no primary key clustered, Dname char(20))

例 5：新建 xk（选课）数据表，并在 Sno（学号）和 Cno（课程号）两个字段上建立表级主键约束 key_xk。

create table xk(sno char(10) not null, cno char(4) not null, constraint key_xk primary key(sno,cno))

同样可以使用 Alter table 语句添加和删除主键约束。

3 UNIQUE 约束

该约束应用于表中的非主键列，UNIQUE 约束保证一列或者多列的试题完整性，确保这些猎不会输入重复的值。例如，表中 UserName 列为主键，但是其中还包括身份证号码列，由于所有身份证号码不可能出现重复，所以可以在此列上建立 UNIQUE 约束，确保不会输入重复的身份证号码。

它与 PRIMARY KEY 约束的不同之处在于，UNIQUE 约束可以建立在多个列之上，而 PRIMARY KEY 约束在一个表中只能有一个。同样，对于一列的 UNIQUE 约束，我们称之为列级 UNIQUE 约束，对于多列的 UNIQUE 约束，我们称之为表级 UNIQUE 约束。下面给出列级 UNIQUE 约束的定义格式：

```
[CONSTRAINT constraint_name]
UNIQUE [CLUSTERED | NONCLUSTERED]
```

使用 UNIQUE 约束的过程中，还需要注意，如果要对允许空值的列强制唯一性。可以允许空值的列附加 UNIQUE 约束，而只能将主键的约束附加到不允许空值的列。但 UNIQUE 约束不允许表中受约束列有一行以上的值同时为空。

例 6：新建 test1（学生）数据表，在 Sno（学号）设置聚簇型主键约束 pk_sno，在 Sname（姓名）字段上设置非空非聚簇型唯一值约束 un_sname。

CREATE TABLE test1
(sno char(10) constraint pk_sno primary key clustered,
sname char(6)　constraint un_sname unique nonclustered not null,
smajor char(16))

4 CHECK 约束

CHECK 约束的主要作用是限制输入到一列或多列中的可能值，从而保证 SQL Server 数据库中数据的域完整性。例如，可以在建立用户使用库时，强制用户的密码在 10 位以上。每个标允许建立多个 CHECK 约束。同样，我们可以为表中的每个列建立约束，每个列可以拥有多个 CHECK 约束，但是如果使用 CREATE TABLE 语句，只能为每个列建立一个 CHECK 约束。如果 CHECK 约束被应用于多列时，他必须被定义为表级 CHECK 约束。

下面给出列级 Check 约束的定义格式：

```
[CONSTRAINT constraint_name]
CHECK (logical_expressin)
```

例 7：新建 test2（选课）数据表，并将 grade 字段设置为缺省值为 0 并可取空值，并在该字段建立检查约束 ck_grade 使其取值在 0 到 100 之间。

CREATE TABLE test2
(sno char(10),cno char(4),
grade numeric(5,1)　DEFAULT 0 NULL　CONSTRAINT　ck_grade
CHECK (grade>=0 AND grade<=100))

5 FOREIGN KEY 约束

FOREIGN KEY 约束为表中的一列或者多列数据提供数据完整性参照。通常是与 PRIMARY KEY 约束或者 UNIQUE 约束同时使用的。

下面给出列级 FOREIGN KEY 约束的定义格式:

```
CREATE TABLE table_name
( field_name datatype
 [ CONSTRAINT constraint_name ]
 [ [ FOREIGN KEY REFERENCES ref_table [ ( ref_column) ]
 [ ON DELETE { CASCADE | NO ACTION } ]
 [ ON UPDATE { CASCADE | NO ACTION } ]
 [ NOT FOR REPLICATION ] [,…n]
```

例 8：新建 test3（选课）数据表，在 Sno 字段上建立引用 Student 表 Sno 字段的外键约束 fk_s_sno，使当 student 表 sno 字段修改时同步更新 test3 表中对应 sno 字段的值；在 Cno 字段上建立引用 Course 表 Cno 字段的外键约束 fk_c_cno。

```
CREATE TABLE test3
(sno char(10) constraint fk_s_sno foreign key references student(sno)
on update cascade,
cno char(4) constraint fk_c_cno foreign key references course(sno),
grade numeric(5,1) )
```

6 Identity Column

可以通过使用标识列（设定处置和增加值,若缺省都默认为 1）为整型字段设定字段值自动添加，格式为 IDENTITY [(seed , increment)]。 标识

例 9：新建 test4（学生）数据表，在 Sno 字段上设置标识功能实现学号自动从 1 开始自动增 1。Create table test4 (sno int identity, sname char(6))

例 10：新建 test5（学生）数据表，在 Sno 字段上设置标识功能实现学号自动从 10001 开始自动增 1。Create table test5 (sno int identity(10001,1), sname char(6))

5.2.4 默认值（Defaults）

当插入一个新行时，若某个列没有明确指定数据值，该列将自动使用指定的缺省值。缺省值－可以是一个常量、一个内置函数、一个表达式或者一个全局变量。一般在创建表时，应使用缺省。

使用 T－SQL 创建‘缺省值’对象和将‘缺省值’对象绑定到字段。

创建‘缺省值‘对象语句为：CREATE DEFAULT default_name AS constraint_expression

将缺省值对象绑定到相应字段语句格式为：
SP_BINDEFAULT　default_name , object_name
解除字段上的缺省值对象语句格式为：SP_UNBINDEFAULT object_name

例 11：创建一个缺省值对象 df_dept，默认值设为‘测绘学院’。
CREATE DEFAULT df_dept AS ‘测绘学院’

例 12：将缺省值对象 df_dept 绑定到 student 表 sdept 字段上。
SP_BINDEFAULT ‘df_sdept’, ‘student.sdept’

例 13：解除 student 表 sdept 字段缺省值对象。
SP_UNBINDEFAULT ‘student.sdept’

5.2.5 规则（Rule）

规则类似于检查约束，因为它们都是限制输入到某个列的值。然而，与检查约束不同的是，检查约束只检查相对简单的值，而规则可以基于条件表达式或者值的列表限制数据值。与检查约束不同的另一点是，每个列只能有一个规则，并且 SQL Server 规则是作为单独的数据库对象存储的。每个列只能有一个规则，而一个规则可以绑定到多个列上。规则还可以应用到用户定义的数据类型上。
提示：在每个列上只能有一个规则，而在同一个列上可以有多个检查约束。常用的办法是，当可以选择时，最好使用约束而不是规则。

使用 T－SQL 创建‘规则’对象和将‘规则’对象绑定到字段。

创建‘规则’对象语句为：CREATE RULE rule_name AS
condition_expression

将‘规则’对象绑定到相应字段语句格式为：
SP_BINDRULE rule_name, object_name

解除字段上的‘规则’对象语句格式为：SP_UNBINDRULE object_name

例 11：创建一个规则对象 rule_age，使其取值范围为 15 到 30 岁之间。
CREATE RULE rule_age AS @value >=15 and @value=<30

例 12：将规则对象 rule_age 绑定到 student 表 sage 字段上。
sp_bindrule ‘rule_age’, ‘student.sage’

例 13：解除 student 表 sage 字段规则对象。
sp_unbindrule ‘student.sage’

5.2.6 触发器（Trigger）

1 触发器的概念

触发器是一种特殊类型的存储过程，与表紧密相连。它是通过事件进触发而被执行的一段 T-SQL 语句，能进行复杂的逻辑处理。当用户操作表中数据时，触发器将自动执行。

触发器的作用：

① 级联修改数据库中的相关表

② 执行比核查约束更为复杂的约束操作

③ 拒绝或回滚违反参考完整性的操作

④ 比较表修改前后数据之间的差别，并根据差别采取相应的操作

触发器的分类：

（1）AFTER 类型触发器

这类触发器将在表中的数据变动（ INSERT 、UPDATE、DELETE）完成以后才被激发，是为了对变动的数据进行检查。如果发现错误，将拒绝或回滚变动的数据（rollback），一个表可以创建多个 AFTER 类型的触发器。

（2）INSTEAD OF 类型触发器

INSTEAD OF 触发器是 SQL SERVER2000 中新增的功能，这种类型的触发器将在数据变动之前被激活，并取代数据的操作（ INSERT 、UPDATE、DELETE ），转而去执行触发器定义的操作。

注意：触发器是针对某一具体操作触发的，所以在定义触发器时必须指定触发操作： INSERT 、UPDATE、DELETE。至少指定一种。如果时 AFTER 触发器可以同时指定多个，如果是 INSTEAD OF 触发器只能指定一个。

2 触发器的工作原理

当向数据表执行 INSERT、UPDATE、DELETE 语句时，若该表设置了触发器，则 SQL SERVER 将根据不同的操作自动生成一个或两个临时表：inserted 表、deleted 表，同时将操作数据送入 inserted 表或 deleted 表。

inserted 表和 deleted 表是 SQL SERVER 为触发器创建的临时表，存储在内存中，不是存储在数据库中，不允许用户直接对其修改。它的表结构和定义触发器相关表的结构相同。触发器工作完成后，与之相关的逻辑表将自动删除。

（1）INSERT 触发器的工作原理

当使用 INSERT 语句向数据表插入一条记录，相关的 INSERT 触发器将自动触发执行。此时，INSERT 触发器自动创建一个 inserted 表，插入的记录被同时添加到 inserted 表和数据表中，触发器检测 inserted 表和数据表用于确定 INSERT 触发器中的操作是否执行。

（2）DELETE 触发器的工作原理

当使用 DELETE 语句向数据表删除记录时，与之相关的 DELETE 触发器将自动触发执行。

此时 DELETE 触发器自动创建一个 deleted 表，删除的记录从数据表中被删除，并放入 deleted 表中。因此，deleted 表和数据表没有相同的行，触发器检测 deleted 表和数据表用于确定 DELETE 触发器中的操作是否执行。

（3）UPDATE 触发器的工作原理

UPDATE 语句相当于在数据表中先执行了 DELETE 操作，后执行了 INSERT 操作，即先删除旧记录，马上插入新记录。当数据表使用 UPDATE 语句修改记录时，UPDATE 触发器将自动触发执行。此时，UPDATET 触发器自动创建一个 inserted 表和 deleted 表，将要删除的记录放入 deleted 表，将新的记录放入 inserted 表。因此，触发器检测 inserted 表、deleted 表和数据表，用于确定是否修改了数据行和 UPDATE 触发器中的操作是否执行。

最后，当确定触发器中的操作是不可执行的，用 rollback 语句撤销所有事务，使数据表回来语句执行前的状态。

3 使用 T-SQL 语言管理触发器

创建触发器的语句格式：

```
CREATE TRIGGER trigger_name ON{table|view}
[WITH ENCRYPTION]
{ FOR | AFTER | INSTEAD OF }{ [UPDATE][,][ INSERT ] [ , ] [DELETE ]}
AS
   sql_statement[,…n]
```

参数说明：

① trigger_name：指定触发器的名称。虽然触发器时基于数据表创建的，但是它在数据库中是唯一的。 ② table|view：创建触发器的表或视图。只有 INSTEAD OF 触发器才能基于视图创建。

③ WITH ENCRYPTION：加密触发器定义语句。和用于视图的加密语句一样，也是不可逆的。

④ FOR | AFTER | INSTEAD OF：指定触发器的类型。如果指定 FOR 关键字和 AFTER 关键字，表示创建的是 AFTER 触发器；如果指定 INSTEAD OF，表示创建的是 INSTEAD OF 触发器。

⑤ [UPDATE][,][INSERT] [,] [DELETE]：指定在表上执行哪些数据修改语句时将激活触发器。必须指定一个选项。允许以任意顺序组合的这些关键字。INSTEAD OF 触发器中每一种操作只能存在一个。

⑥ sql_statement：定义触发器的语句（文本）。指定过程要执行的操作。

例 14：在 student 表上创建一个 after 类型的名称为 student-update 的触发器，实现当更新 student 表里的学号时，同时去级联更新这个同学在选课

表里对应的学号，并给出修改结果信息。

```
CREATE TRIGGER student_update ON student FOR UPDATE
AS
IF update(sno)
  begin
     update sc set sno=(select sno from inserted)
                    where sno=(select sno from deleted)
     print ‘级联修改成功！’
  end
ELSE
   print ‘没有对应的选课记录！’
```

5.3 实验内容

要求使用企业管理器和 SQL 语句来完成如下内容。

注意：如果创建过程中提示数据库中已存在同名对象，请自行更换相应对象名称。

（1）在查询分析器中使用 CREATE TABLE 语句，在 JXGL 数据库中创建符合下表中完整性约束条件的学生表 S。

列名	数据类型	能否空值	默认值	键/索引	说明
SNO	CHAR(6)	否		主键、聚集索引	学号
NAME	CHAR(8)	否		唯一约束	姓名
AGE	NUMERIC(2)				年龄
SEX	CHAR(2)		'男'		性别
DEPT	CHAR(10)				所在系

（2）在查询分析器中使用 CREATE TABLE 语句，在 JXGL 数据库中创建符合下表中完整性约束条件的选课表 temp。

列名	数据类型	能否空值	检查	键/索引	说明
SNO	CHAR(6)	否		组合主键、聚集索引 外键 student(sno)	学号
CNO	CHAR(4)	否		组合主键、聚集索引 外键 course (cno) 主表更新采用级联	课程号
SCORE	NUMERIC(2)		0~100		成绩

注：组合主键、聚集索引定义在 SNO 和 CNO 上；外键上还需定义一个非聚集索引。

（3）使用企业管理器为 JXGL 数据库创建一个 age_rule 规则，并将其绑定到学生表 student 的 Sage 列，使 Sage 在 10 到 50 之间取值。

（4）使用查询分析器为 JXGL 数据库创建一个 sex_rule 规则，并将其绑定到

学生表 Student 的 Ssex 列，使 Ssex 只能取值为’男’或’女’。

（5）使用企业管理器为 JXGL 数据库创建一个 dept_default 默认，并将其绑定到学生表 Student 的 Sdept 列，设置该列的默认值为’测绘学院’。

（6）使用企业管理器为 JXGL 数据库创建一个数据表，表名、列名和数据类型自定，给其中一个字段设置为标识列，并指明初始值和增加量（自定）。

（7）使用企业管理器为 JXGL 数据库创建一个数据表 Test，列名和数据类型自定，利用设计表窗口和属性对话框创建和修改约束，掌握企业管理器建立主键约束、外键约束和检查约束等方面的技能。

（8）使用 T-SQL 语句管理触发器

① 在 course 表中创建一个触发器，禁止更新或者添加数据的操作。

② 在表 TEACHERS 中创建 AFTER 类型触发器 trigger_change ，限制每次工资 (pay) 的变动不能超过 2000 。注意：通过比较 inserted 表和 deleted 表的 pay 来判断。

在查询分析中输入下列语句。

```
update teachers
  set pay=6000
  where tno='001'
```

问题： 该语句执行结果是什么？查看数据表该条记录是否被修改 ？ 如果没有被修改，分析其原因。

③ 在 TEACHERS 表中创建一个 AFTER 类型触发器 trigger_del ，监控删除的教师记录。（如果被删除的教师是 “ 讲师 ” 职称，则提示不能删除这个教师，否则可以删除）注意：根据 deleted 表中的记录来判断。在创建完触发器后，在查询分析中输入下列语句，验证触发器是否能正常工作。

```
delete teachers where tno='030'
```

④ 在 teachers 表上建立一个 AFTER 类型的触发器，监控对老师工资的更新，当更新后的工资比更新前小时，取消操作，并给出提示信息，否则允许。

思考题

（1）SQL Server 实现数据完整性有哪两种方法?

（2）SQL Server 有几种约束类型？它们分别是什么?

（3）简述 SQL Server 触发器的工作原理?

上机实验 6　　用户管理和权限管理

6.1 实验目的

1、理解数据库安全性的概念和原理；

2、掌握创建用户的方法和分配权限的方法。

6.2 实验练习预备知识点

6.2.1 设置 SQL Server 的安全认证模式

1、在对象资源管理器中展开 SQL 服务器组。用鼠标右键单击需要设置的 SQL 服务器，在弹出的菜单中选择“属性”项，如图 6－1 所示。

2、在弹出的 SQL 服务器属性对话框中，选择“安全性”选项卡，如图 6－2 所示。

3、在安全性选项卡中有一个安全性栏，它包括两个单选钮：单击“SQL Server 和 Windows(S)为选择混合安全认证模式；单击“仅 Windows(W)”则为选择集成安全认证模式。

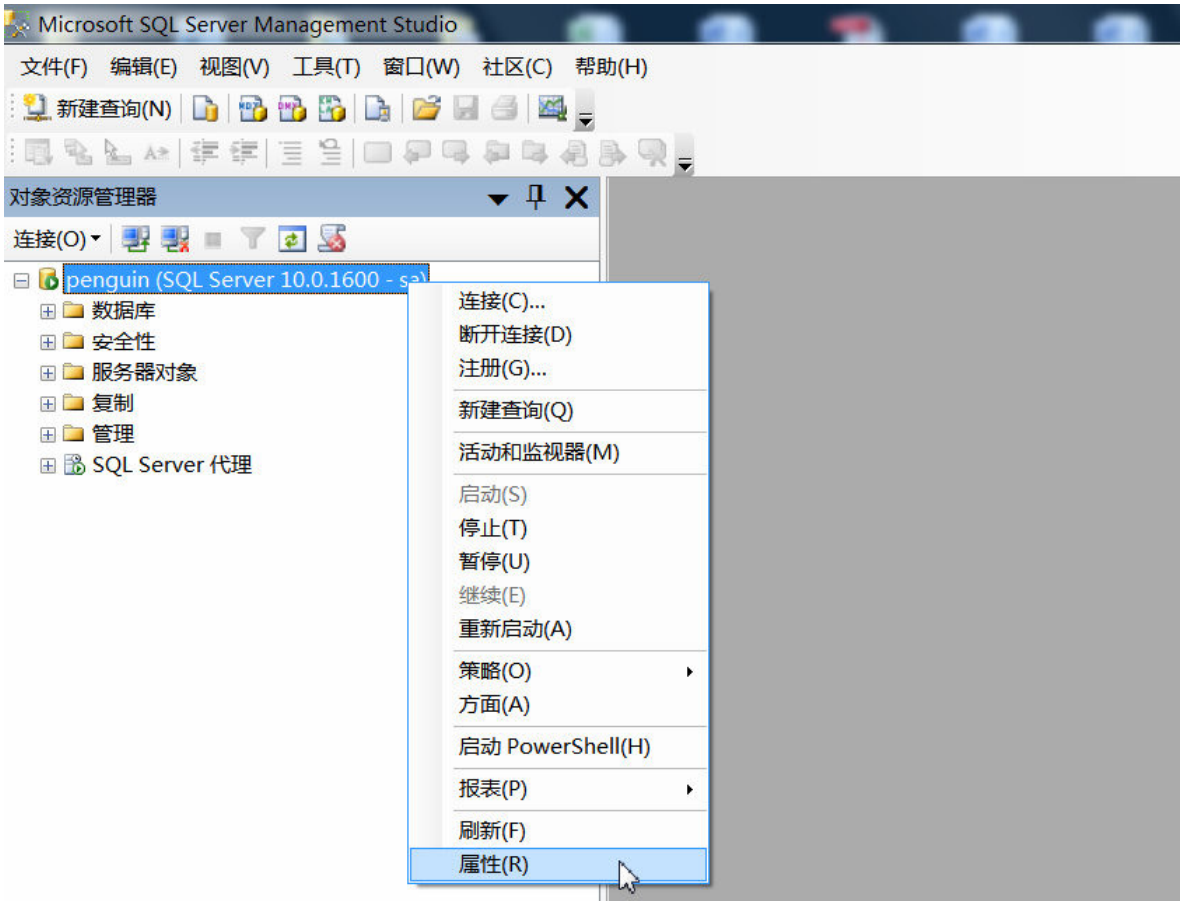


图 6-1 SQL 服务器的弹出菜单

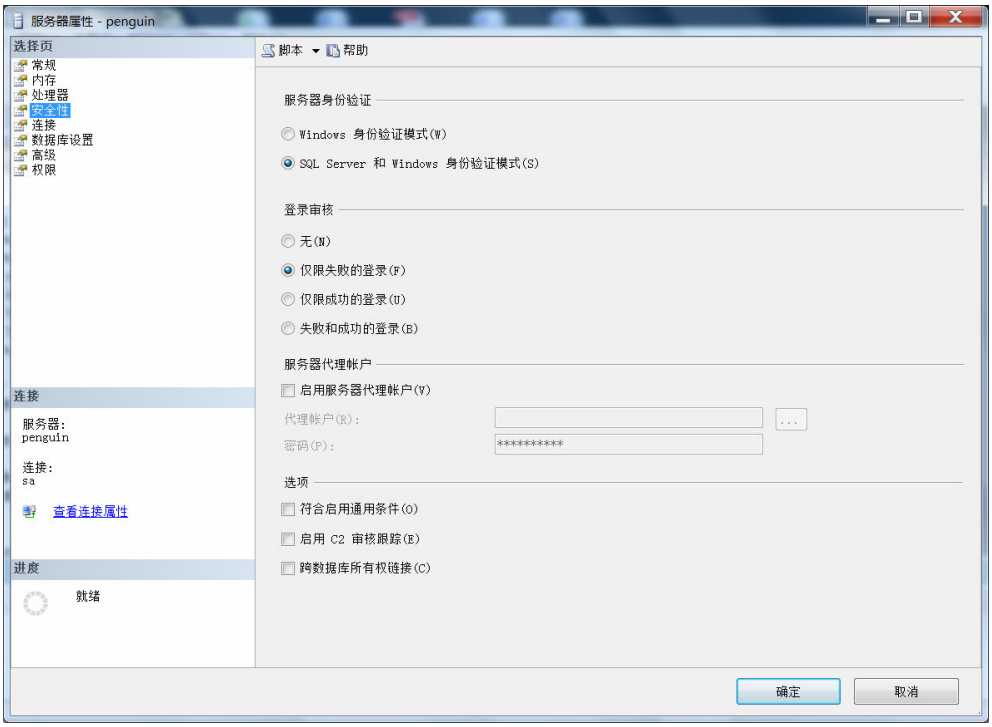


图 6－2　SQL Server 属性的安全性页面

6.2.2 登录的管理

1、查看安全性文件夹的内容

使用对象资源管理器可以创建、查看和管理登陆，登录文件夹存放在 SQL 服务器的安全性文件夹中。当执行了进入对象资源管理器，打开指定的 SQL 服务器组和 SQL 服务器，并选择安全性文件夹的系列操作后，就会出现如图 6－3 所示的屏幕窗口。

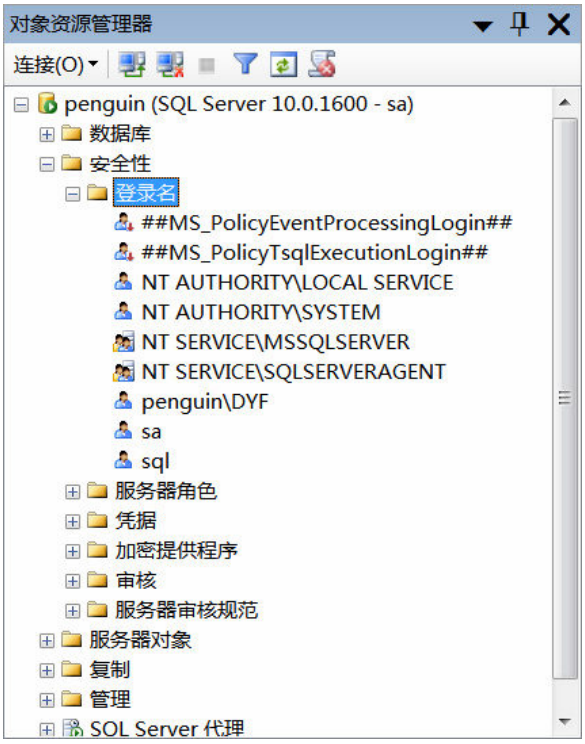


图 6－3 SQL server 的安全性文件夹

通过该窗口可以看出，安全性文件夹包括登录文件夹、服务器角色文件夹、链接服务器文件夹和远程服务器文件夹等。其中：登录文件夹用于存储和管理登录用户，服务器角色文件夹用于存储和管理角色：链接服务器文件夹用于存储和管理连接的服务器．远程服务器文件夹用于存储管理远程服务器信息。

2、创建一个登录用户

① 用鼠标右键单击登录文件夹，出现如图 6－4 所示的弹出菜单、在弹出的菜单中选择“新建登录”选项后 就会出现一个登录属性对话框．如图 6－5 所示。

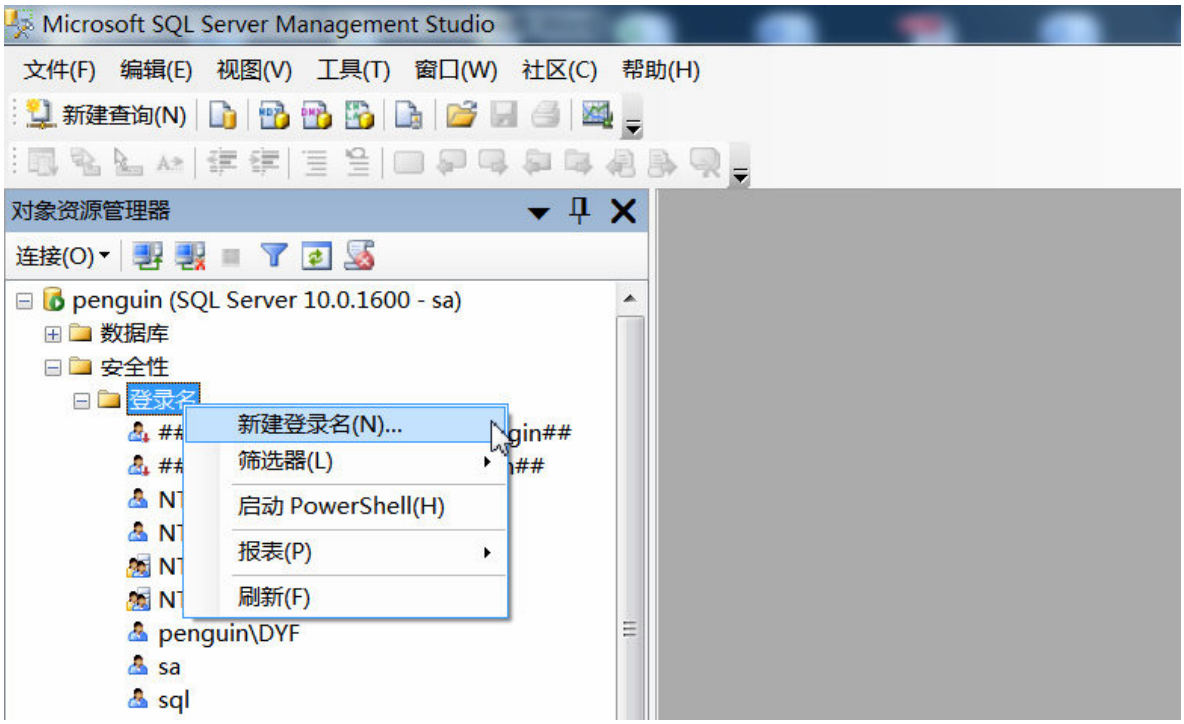


图 6－4 登录文件夹的弹出菜单

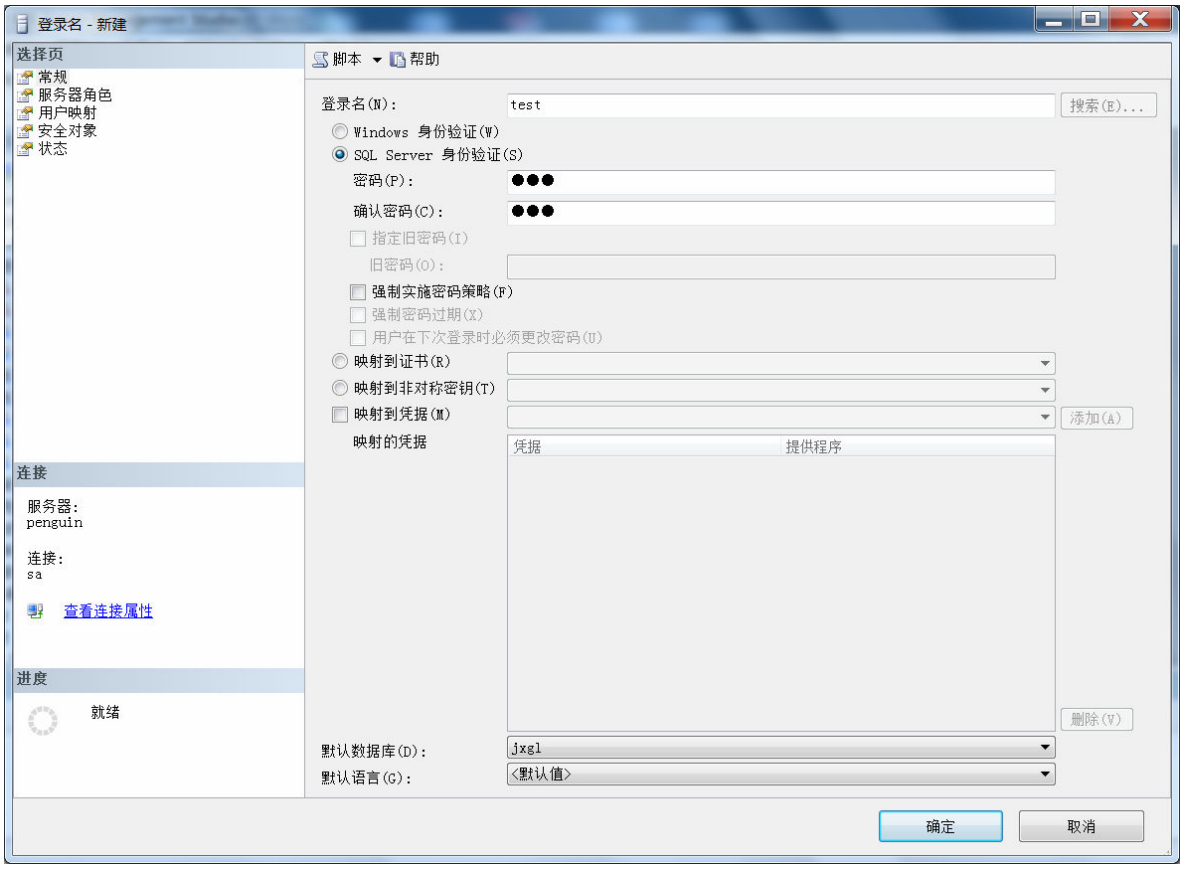


图 6－5 登录对话框中的常规页面

② 选择常规选项卡输入用户的一般特征、常规选项卡界面如图 6－5。所示，在

常规选项卡中要输入用户名。选择该用户的安全认证模式。选择默认数据库和默认语言。如果选择 Windows 身份验证，需要单击名称右边的“….”按钮，调出 windows 已有的登录用户，如图 6－6 所示，从中选择新建的登录名称。

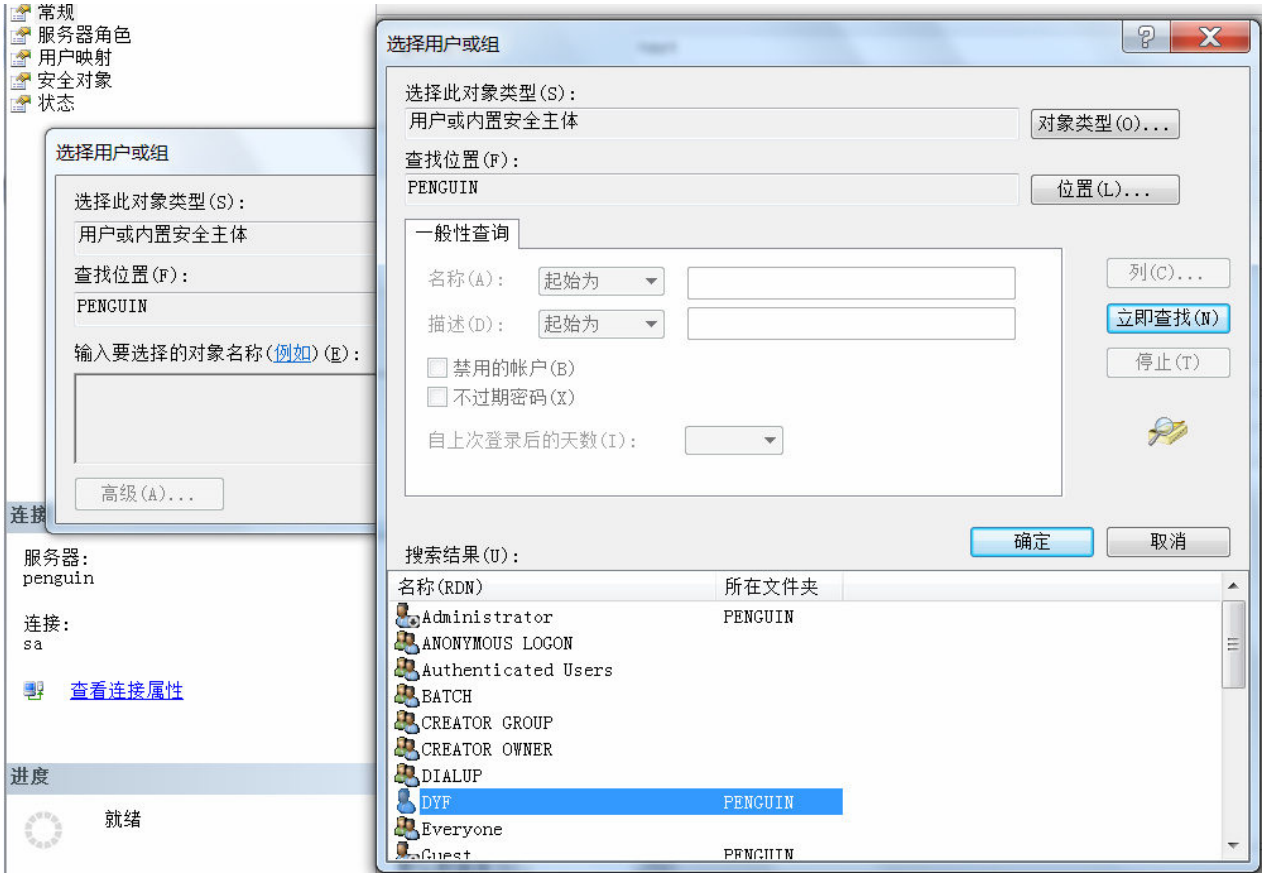


图 6－6 Windows 系统具有的默认登录用户

③ 选择服务器角色选项卡，确定用户所属服务器角色。服务器角色选项卡如图 6-7 所示，服务器角色选项卡在的服务器角色列表中列出了系统的固定服务器角色，在这些固定服务器角色的左端有相应的复选框 ，该登录用户就成为相应的服务器角色成员了。在下面描述栏目中，列出了当前被选中的服务器角色的权限。

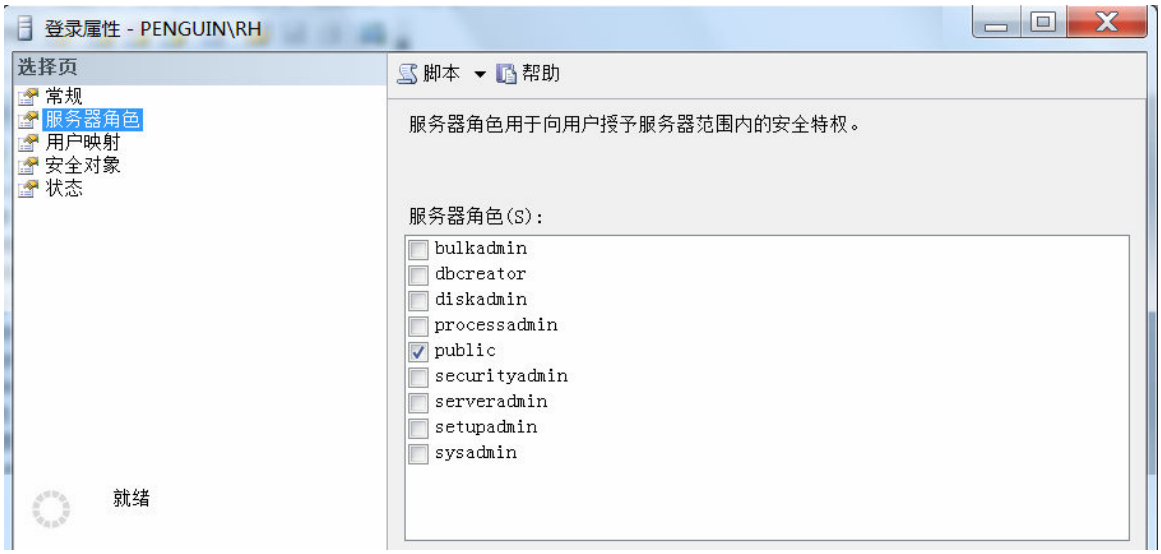


图 6－7 登录服务器角色选项卡

④ 选择用户映射选项卡，确定用户能访问的数据库，并确定用户所属的数据库角色。用户映射选项卡界面如图 6－8 所示，在用户映射选项卡中有两个列表框，上面的列表框中列出了 SQL 服务器全部的数据库，单击某个数据库左端的复选框， 表示允许该登录用户访问相应的数据库，他右边为该登录用户在数据库中使用的用户名可以对其进行修改；下面为当前被选种数据库的数据库角色清单，单击某个数据库角色左端复选框，表示是该登录用户成为它的一个成员。

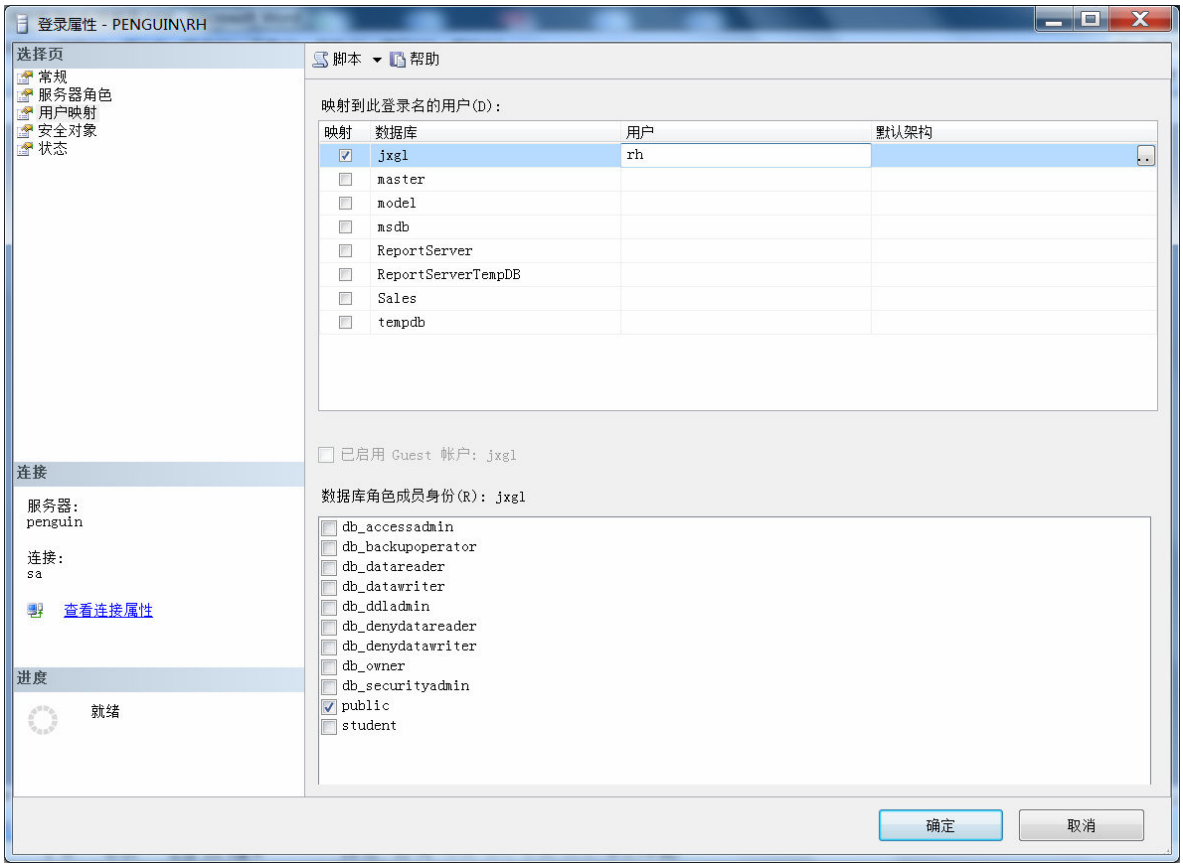


图 6－8 新建登录的数据库访问选项卡

⑤操作完成后，单击“确定”按钮，即完成了创建登录用户的工作。

6.2.3 数据库用户的管理

登录用户只有成为数据库用户（Database User）后才能访问数据库。每个数据库的用户信息都存放在系统表 Sysusers 中，通过查看 Sysusers 表可以看到该数据库所有用户的情况。SQL Server 的任一数据库中都有两个默认用户：dbo(数据库拥有者用户)和 guest（客户用户）。通过系统存储过程或对象资源管理器可以创建新的数据库用户。

1、dbo 用户:dbo 用户即数据库拥有或数据库创建者，在其所拥有的数据库中拥有所有的操作权限。Dbo 的身份可被重新分配给另一个用户，系统管理员 Sa 可以做为他所管理系统的任何数据库的 dbo 用户。

2、guest 用户:如果 guest 用户在数据库存在，则允许任意一个登录用户作为 guest 用户访问数据库，其中包括那些不是数据库用户的 SQL 服务器用户。除系统数据库 master 和临时数据库 tempdb 的 guest 用户不能被删除外，其它数据库都可以将自己的 guest 用户删除，以防止非数据库用户的登录用户对数据库进行访问。

3、创建新的数据库用户

要在 JXGL 数据库中创建一个“User1”数据库用户，可以按下面的步骤创建新数据库用户。① 在对象资源管理器中展开 SQL 服务器及目标数据库下“安全性”文件夹。用鼠标右键单击用户文件夹，弹出一个快捷菜单，如图 6－9 所示，在弹出的菜单中选择“新建数据库用户”项，会出现如图 6－10 所示的新建数据库用户属性对话框。

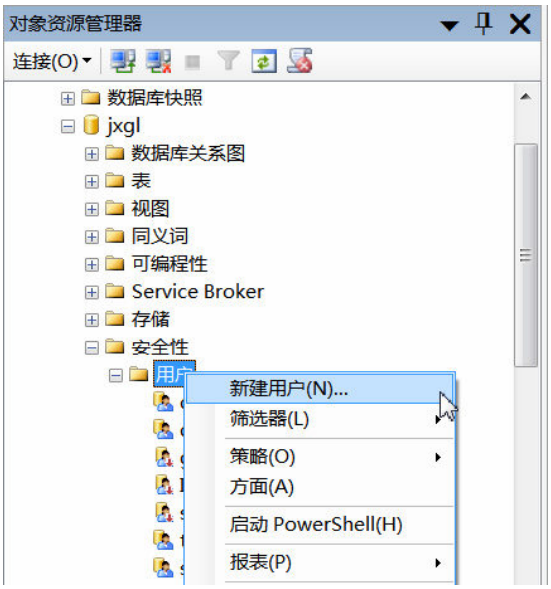


图 6－9 新建数据库用户的弹出菜单

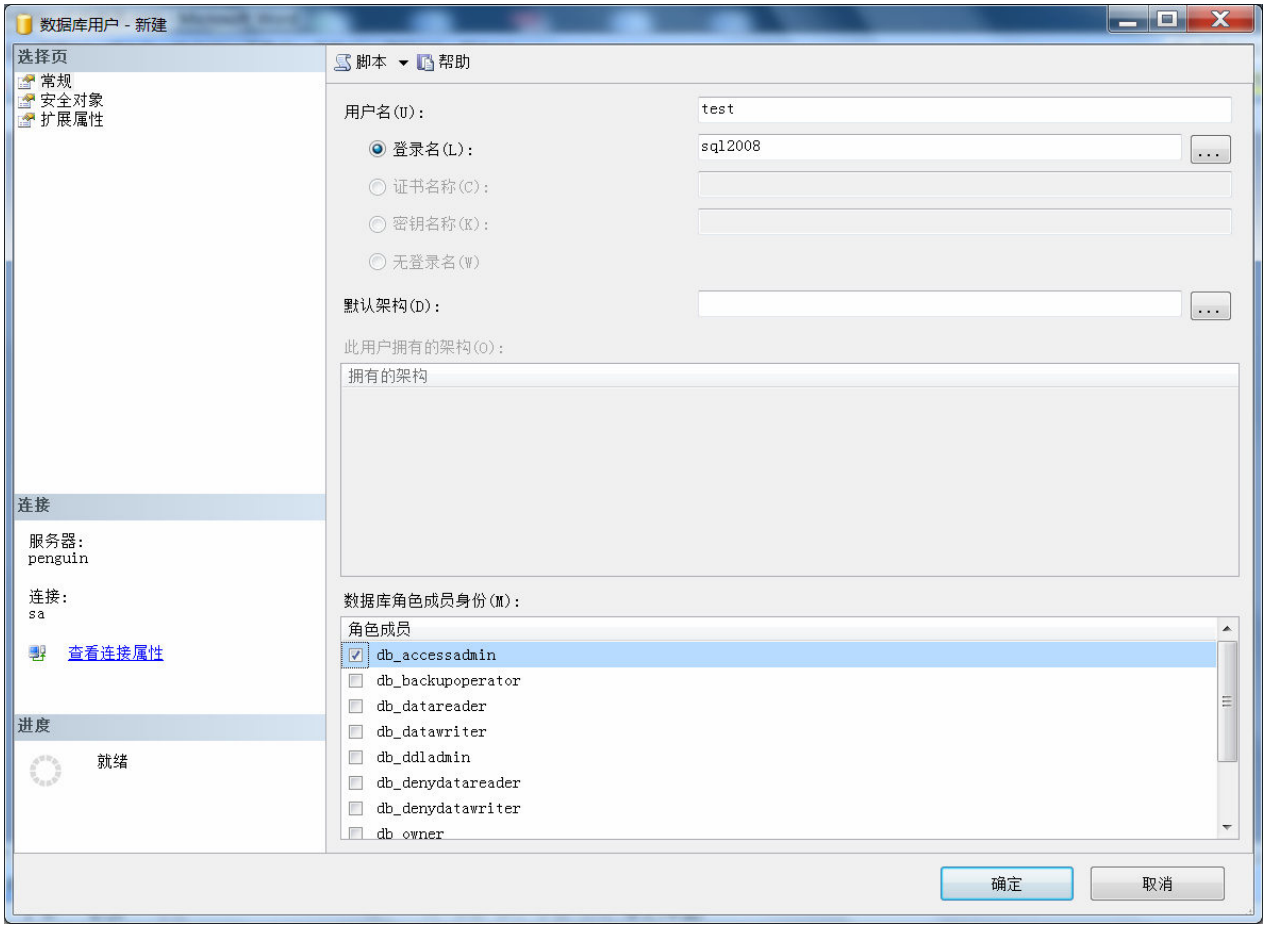


图 6－10 新建数据库用户属性对话框

② 对话框的登录名栏中选择一个 SQL 服务器登陆账号名，本例为“sql2008”，并在他下面的用户名栏中输入数据库用户参加的角色。

③ 单击“确定”按钮。

6.2.4 服务器及角色的管理

登录用户可以通过两种方法加入到服务器角色中：一种方法是在创建登录时，通过服务器角色页面中的服务器角色选项，确定登录用户应属于的角色；另一种方法是对已有的登录，通过参加或移出服务器角色的方法。

使登录用户加入服务器角色的具体步骤为：

① 在对象资源管理器中指定的 SQL 服务器、安全性文件夹、单击服务器角色后，就会在右面的细节窗口中出现 8 个预定义的服务器级角色，如图 6－10 所示。

② 选中一个服务器级角色，例如 Database Creators，右击，弹出菜单如图 6－11 所示。

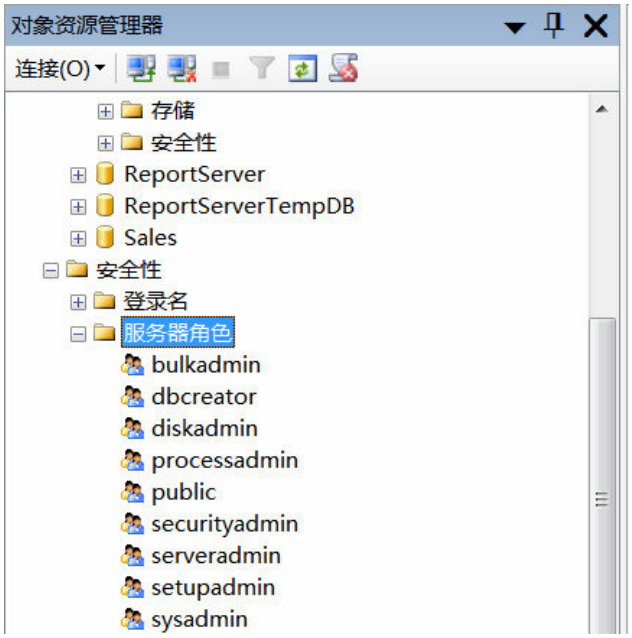


图 6－10　SQL server 的服务器级角色

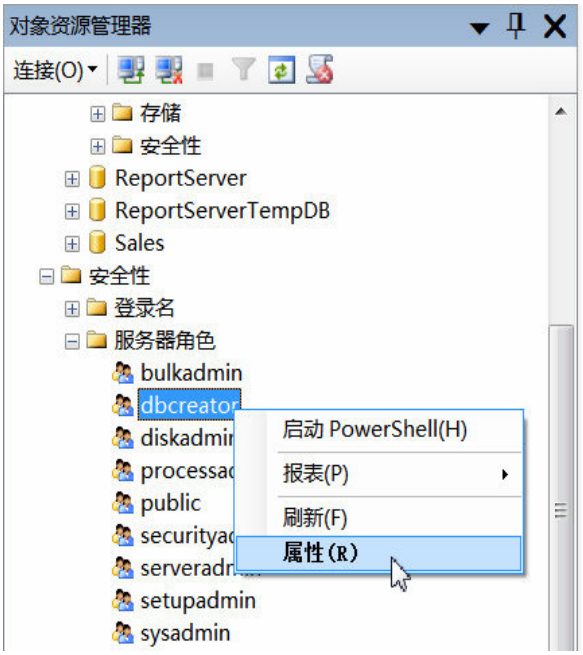


图 6－11 服务器角色的弹出菜单

③ 在弹出的菜单中选择“属性”项后，就会出现一个服务器角色属性对话框，如图 6－12 所示。在服务器角色对话框中，将登录用户添加到服务器角色中或从服务器角色中移去登录用户；并单击“添加”按钮，在出现的选择登录用户对话框中，选择登录名后。单击“确定’”按钮，之，新选的登录就会出现在常规对话框中。如果要从服务器角色中移去登录，则先选中登录用户，再单击“删除”按钮即可。

图 6－12 服务器角色属性的常规页面

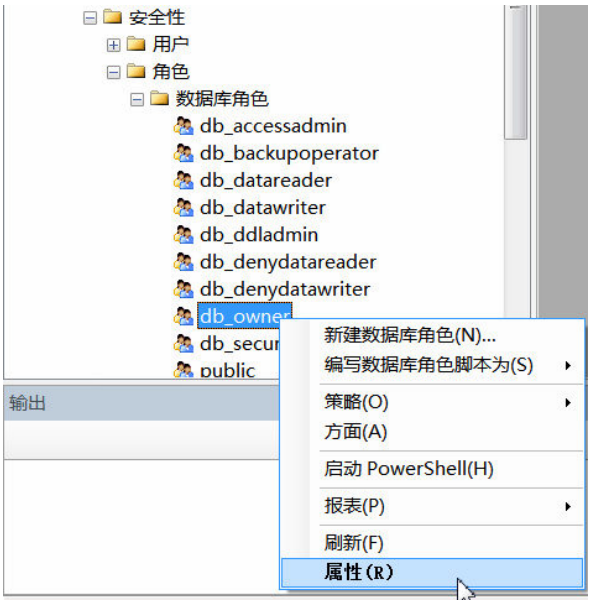


图 6-13 数据库角色的弹出菜单图

6.2.5 数据库角色的管理

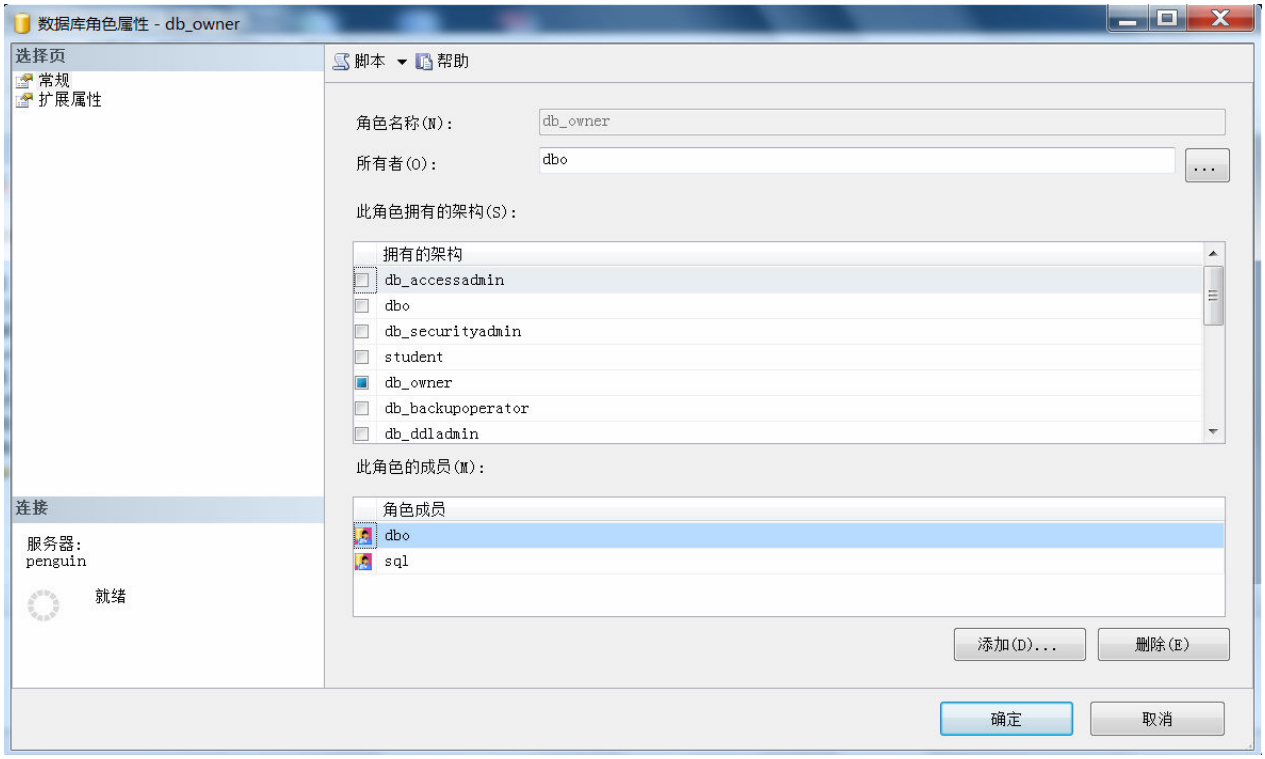
1、在数据库角色中增加或移去用户

在对象资源管理器中，向数据库角色添加或移去用户的方法是:

① 展开一个 SQL 服务器、数据库文件夹和指定的数据库文件夹下的“安全性”文件夹，选中角色文件夹后，在细节窗口中就会出现该数据库已有的角色。

② 选中要加入的角色。例如选中 db-owner 角色，用鼠标右击它,在弹出的菜单中选择“属性”项，如图 6－13 所示。

③ 在如图 6－14 所示数据库角色属性对话框中，单击“添加”按钮，则出现选择该数据库用户的对话框，选择要加入角色的用户，单击“确定”按钮，关闭选择数据库用户对话框后，会发现新选的用户名出现在数据库角色属性对话框中。



6-14 数据库角色属性对话框

④ 如果在数据库角色中要移走一个用户，在用户栏中选中它后，单击“删除”按钮。

⑤ 完成后，单击“确定”按钮。

2、创建新的数据库角色

① 在对象资源管理器中打开 SQL 服务器组、服务器、数据库文件夹和特定的数据库文件夹的“安全性”文件夹.

② 用鼠标选中角色子文件夹后。右边的细节窗口显示该数据库中的角色，用鼠标右击任意角色，并在弹出的菜单中选择“新建数据库角色”项 如图 6-15 所示。

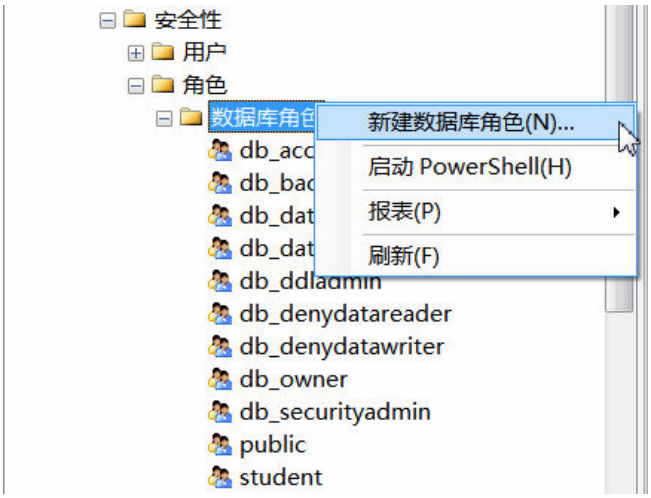


图 6-15 新建数据库角色选项

③ 在如图 6－16 所示的新建数据库角色对话框的名称栏中输入新角色名；在架构列表框中选择继承已有数据库角色的权限，在成员栏中增加或移去角色中的用户。

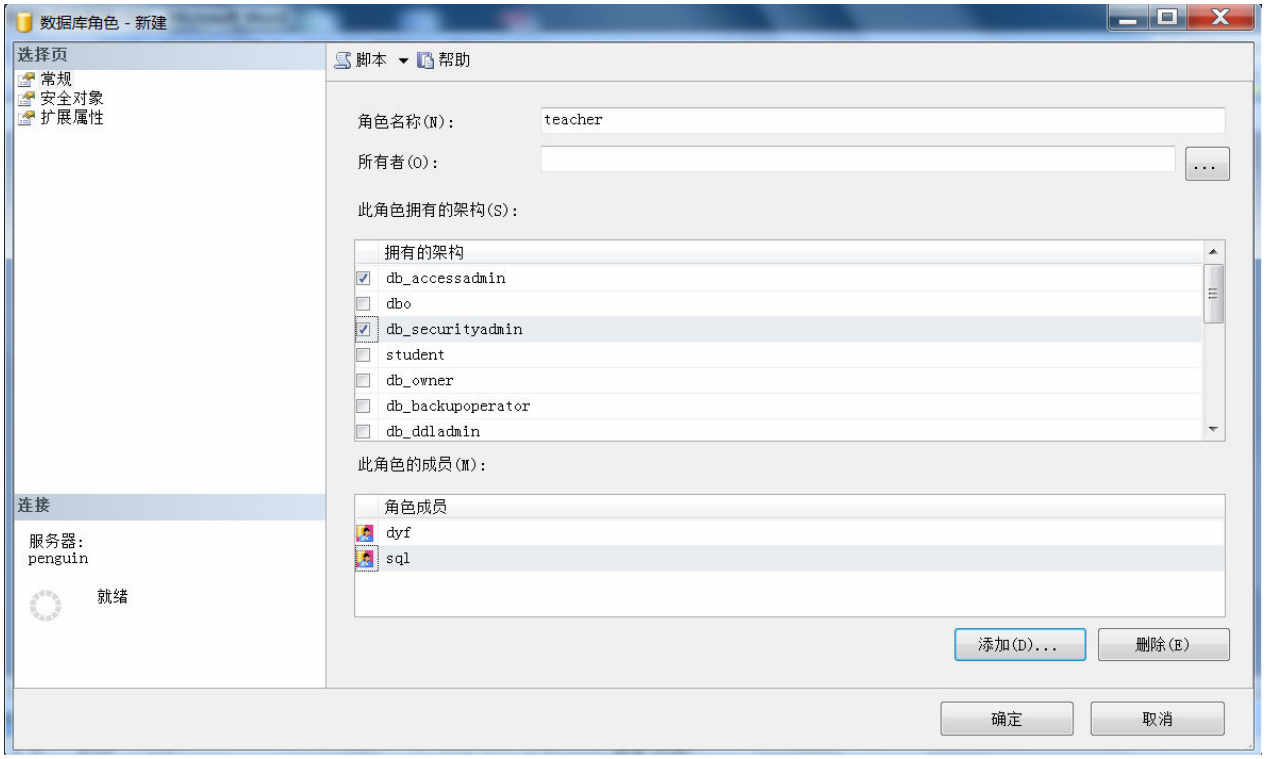


图 6-16 新建数据库角色对话框

④ 单击“确定”按钮完成。

6.2.6 对象权限的管理

对象极限的管理可以通过两种方法实现：一种是通过对象管理它的用户及操作权；另一种是通过用户管理对应的数据库对象及操作权。具体使用哪种方法要视

管理的方便性来决定。

1、通过对象授予、撤消和废除用户权限

如果要一次为多个用户（角色）授予、撤消和废除对某一个数据库对象的权限时。应采用通过对象的方法实现。在 SQL Server 2008 的对象资源管理器中，实现对象权限管理的操作步骤如下：

① 展开 SQL 服务器、数据库文件夹和数据库，选中一个数据库对象。例如，选中 JXGL 数据库中的表文件夹中的 student 表。单击鼠标右键，使之出现弹出菜单。

② 在弹出的菜单中，选择“属性”项，如图 6－17 所示。在弹出的表属性对话框中选择“权限”页，如图 6-18 所示。

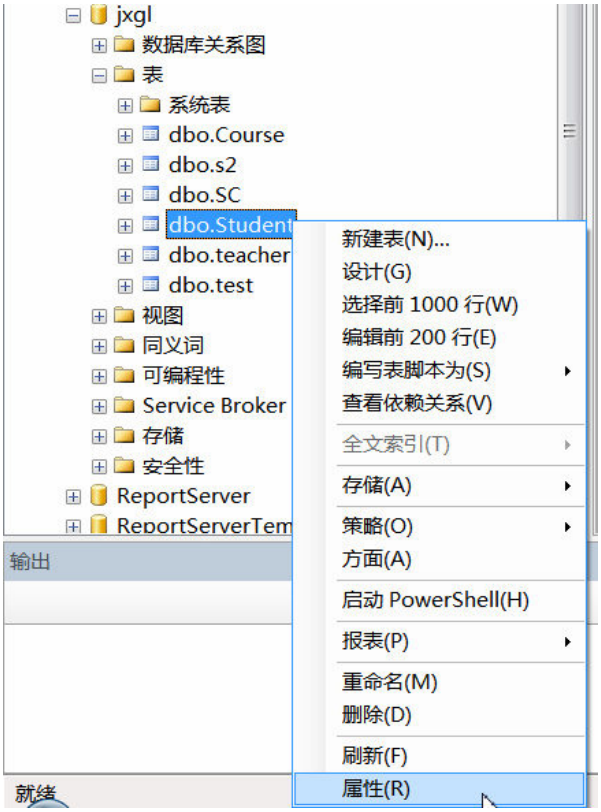


图 6－17 在对象的弹出菜单中选择属性项

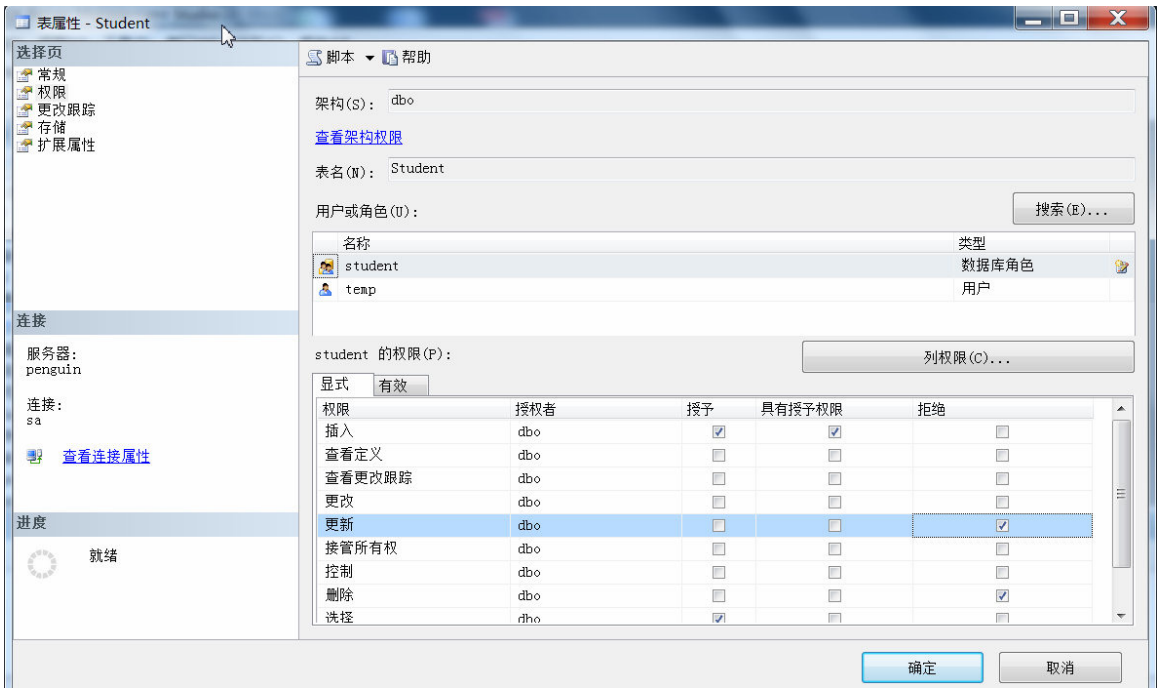


图 6－18 对象权限对话框

③ 在对象权限对话框右边有两个列表框：上面是“用户或角色”列表，下面是选中对应的用户或角色对该表对象所拥有的操作权限列表。

④ 在对象权限对话框的下面是有关数据库用户和角色所对应的权限表，这些权限均以复选框的形式表示。复选框有三种状态。“√”为授权：“╳”为废除权；空为撤权。在表中可以对各用户或角色的各种对象操作权（SELECT、INSERT、UPDATE、DELETE、EXEC 和 DRI）进行授予或撤消。

⑤ 完成后单击“确定”按钮。

2、通过用户或角色授予、撤消和废除对象权限

如果要为一个用户或角色同时授予、撤消或者废除多个数据库对象的使用权限，则可以通过用户或角色的方法进行。例如，要对图书_读者数据库中的 roles1 角色进行授权操作。

在对象资源管理器中，通过用户或角色授权（或收权）的操作步骤如下：

① 展开一个 SQL 服务器和目标数据库下的“安全性”文件夹，单击用户或角色文件夹。在细节窗口中找到要选择的用户或角色，本例为角色中的 dyf 用户，用鼠标右键单击 dyf 用户。在弹出的菜单中选择属性项后，出现形式如图 6-19 所示的数据库用户属性对话框。

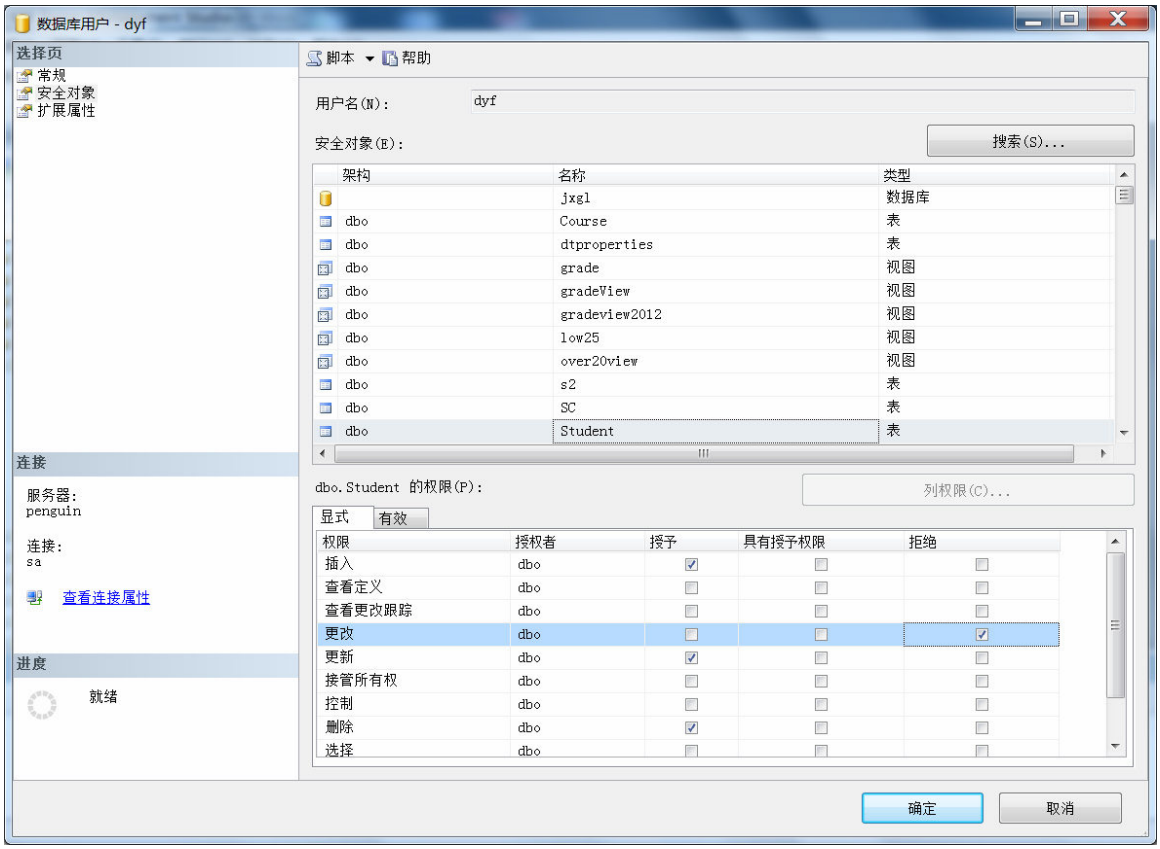


图 6-19 数据库角色属性对话框

② 在数据库用户属性对话框中，中间的列表框列出该数据库中全部对象；下面的列表框显示该用户对上面某个选定的对象所具有的操作权限。

③ 在对话框中的权限列表中，对每个对象进行授权、撤消权和废除极限操作。在权限表中，权力 SELECT、INSERT、UPDATE 等安排在列中，每个对象的操作权用一行表示。在相应的单元格中，如果为“√”表示授权；“╳”表示废除权限；空白表示撤消权力。单击单元格可改变其状态。

④ 完成后，单击“确定”按钮。

6.2.7 语句权限的管理

在 SQL Server 2008 的对象资源管理器中，还提供了管理语句权限的方法，其操作的具体步骤如下：

① 展开一个 SQL 服务器、数据库文件夹，用鼠标右键单击指定的数据库文件夹。例如，JXGL 数据库，在弹出的菜单中选择“属性”项，如图 6－20 所示。会出现数据库属性对话框。

② 在数据库属性对话框中，选择“权限”选项卡，出现管理数据库语句权限的对话框，如图 6-21 所示。在对话框的列表栏中，单击表中的各复选小方块可分

别对各用户或角色授予、撤消和废除数据库的语句操作权限。小方框内的“√”表示授予权限；“╳”表示废除权限；空白表示撤消权限。

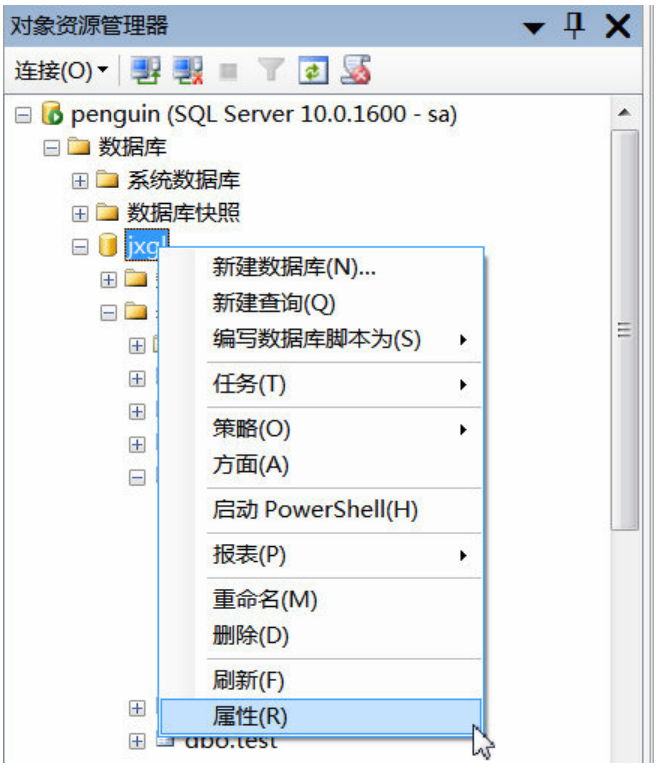


图 6－20 数据库的弹出菜单

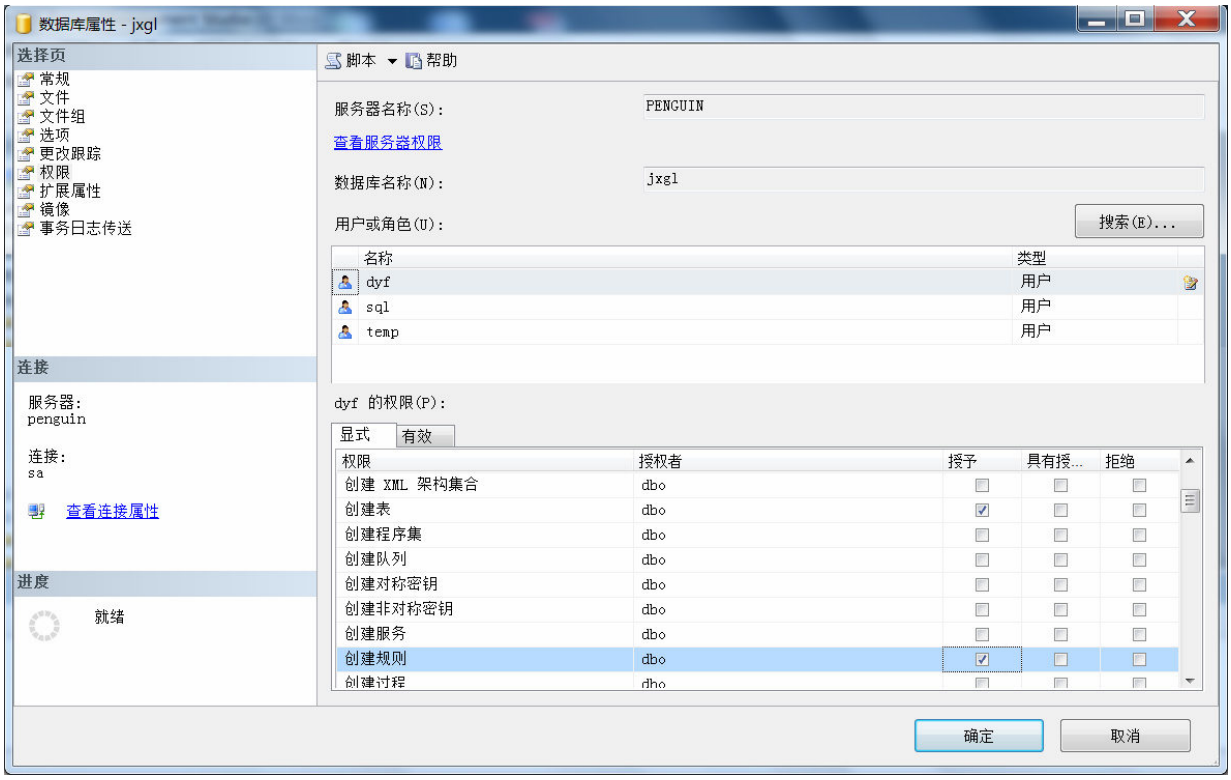


图 6－21 管理数据库语句权限对话框

③ 完成后单击“确定”按钮。

6.2.8 相关用户和权限管理语句

1、创建和删除 SQL Server 登录帐户

创建：**sp_addlogin** [**@loginame =**] '*login*'
[, [**@passwd =**] '*password*']
[, [**@defdb** =] '*database*']
[, [**@deflanguage =**] '*language*']
[, [**@sid =**] *sid*]
[, [**@encryptopt =**] '*encryption_option*']

删除：**sp_droplogin** [**@loginame =**] '*login*'

例 1：创建一个登录名为 templogin 得帐户，密码为 123，默认操作数据库为 JXGL。

sp_addlogin ‘templogin’,@passwd= ‘123’,@defdb = ‘JXGL’

例 2：将 templogin 的登录帐户删除。

sp_droplogin ‘templogin’

2、创建和删除 Database 用户

创建：**sp_grantdbaccess** '*login*' [,’ name_in_db’]

删除：**sp_revokedbaccess** **‘user_name’**

例 3：在 JXGL 数据库中使用 templogin 登录帐户创建一个数据库用户 student。

Use JXGL

sp_grantdbaccess ‘templogin’,’ student’

例 4：将 JXGL 数据库中的用户 student 删除。

sp_grantdbaccess ‘student’

3、添加和删除服务器角色成员

添加：**Sp_addsrvrolemember** **[@loginame=] ‘login’,[@rolename=] ‘role’**

删除：**Sp_dropsrvrolemember** **[@loginame=] ‘login’ ,[@rolename=] ‘role’**

例 5:将 templogin 帐户添加为 sysadmin 角色成员。

Sp_addsrvrolemember ‘templogin’, ‘sysadmin’

例 6:将 templogin 帐户从 sysadmin 角色中删除。

Sp_dropsrvrolemember ‘templogin’, ‘sysadmin’

4、创建和删除数据库角色，添加和删除数据库角色成员

创建数据库角色：**Sp_addrole [@rolename=] ‘role’**

删除数据库角色：**Sp_droprole [@rolename=] ‘role’**

添加数据库角色成员：**sp_addrolemember [@rolename =] 'role' , [@membername =] 'security_account'**

删除数据库角色成员：**sp_droprolemember [@rolename =] 'role' , [@membername =] 'security_account'**

注：添加角色成员可以向系统固定的数据库角色添加成员，如 db_owner.

例 7：在 JXGL 数据库中先创建一个角色 temprole，并将 user1 用户添加为该角色成员，然后将角色成员删除，最后删除 temprole 角色。

use jxgl

Sp_addrole ‘temprole’

Sp_addrolemember ‘temprole’,’user1’

Sp_droprolemember ‘temprole’,’user1’

Sp_droprole ‘temprole’

4、授予和撤销对象权限

授权：**GRANT**

```
{ ALL [ PRIVILEGES ] | permission [ ,...n ] }
{ [ ( column [ ,...n ] ) ] ON { table | view }
    | ON { table | view } [ ( column [ ,...n ] ) ]
    | ON { stored_procedure | extended_procedure }
    | ON { user_defined_function } }
  TO security_account [ ,...n ]
  [ WITH GRANT OPTION ]
```

撤销：**REVOKE** [GRANT OPTION FOR]
{ ALL [PRIVILEGES] | *permission* [,...*n*] }
{ [**(** *column* [,...*n*] **)**] ON { *table* | *view* }
| ON { *table* | *view* } [**(** *column* [,...*n*] **)**]
| ON { *stored_procedure* | *extended_procedure* }
| ON { *user_defined_function* } }
{ TO | FROM }
security_account [,...*n*]

禁止：**DENY**
{ ALL [PRIVILEGES] | *permission* [,...*n*] }
{ [**(** *column* [,...*n*] **)**] ON { *table* / *view* }
| ON { *table* | *view* } [**(** *column* [,...*n*] **)**]
| ON { *stored_procedure* | *extended_procedure* }
| ON { *user_defined_function* } }
TO *security_account* [,...*n*]
[CASCADE]

注：这里 PRIVILEGES 包括：select，delete，update，insert，references；All 代表以上所有操作权限。security_account 可以是用户名，角色名，或者 public。

举例：

授予 u1 用户对 student 表的查询权。

grant select on student to U1

将对 Course 表所有权限同时授予给 U2 和 U3 用户。

grant all privileges on course to U2,U3

将对 SC 表的查询权授予给所有用户。

grant select on sc to public

将对 student 表的查询权以及对 Sno 字段的更新权授予给 u4 用户。

grant update(Sno),select on student to U4

将对 SC 表插入数据的权限授给 U5 用户，同时 U5 可以将这种权限授权给其他用户。

grant insert on sc to U5 with grant option

撤销 u4 用户对 student 表 Sno 字段的更新权。

revoke update(Sno) on student from U4

撤销所有用户对 SC 表的查询权。

revoke select on sc from public

撤销 U5 用户对 SC 表的查询权，同时撤销 U5 授给其他用户的这种权限。

revoke select on sc from U5 cascade

拒绝 U1 用户对 student 表的删除权。

DENY　DELETE ON stuent TO u1

5、授予和撤销语句权限

授权：**GRANT** { ALL | *statement* [,...*n*] } TO *security_account* [,...*n*]

撤销：**REVOKE** { ALL | *statement* [,...*n*] } FROM *security_account* [,...*n*]

禁止：**DENY** { ALL | *statement* [,...*n*] } TO *security_account* [,...*n*]

注：这里的 statement（语句权限）包括：create table，create view，create rule，create database， create default， create function， create procedure， backupdatabase，backlog。security_account 可以是用户名，角色名，或者 public。

举例：

授予用户 U1 创建数据库和创建数据表的权限。

GRANT CREATE DATABASE, CREATE TABLE TO u1

撤销用户 U1 创建数据库和创建缺省的权限。

REVOKE CREATE TABLE, CREATE DEFAULT FROM u1

禁止用户 U1 创建数据库和创建数据表的权限。

DENY CREATE DATABASE, CREATE TABLE TO u1

6.3 实验内容

要求使用对象资源管理器、系统存储过程和 SQL 语句来完成如下内容。

（1）使用对象资源管理器在你的 SQL Server 中创建一个登录名 teacher，且

① 它使用 SQL Server 认证；

② 能够创建和修改数据库；

③ 能访问 pubs 数据库和 JXGL 数据库；

④ 并且能够在这些库中创建数据表、视图、规则、缺省的语句权限；

⑤ 对 JXGL 数据库中的 student 表具有插入、修改和删除的权限。

（2）在查询分析器中使用系统存储过程和 SQL 语句完成下列任务：

① 创建一个登录名 student，口令为 123，缺省数据库为 Northwind；

② 将其加入到 JXGL 数据库的用户中（用户名自己给定）；

③ 将其加入到服务器的 sysadmin 角色中；

④ 将其加入到 JXGL 数据库的 db_owner 角色中；

⑤ 授予他在 JXGL 数据库中创建视图、创建表的权限；

⑥ 授予他对 JXGL 数据库中的 student 表具有所有权限，且可将这些权限授予他人；

⑦ 撤销他对 JXGL 数据库中的 student 表的修改权限；

⑧ 禁止他对 JXGL 数据库中的 student 表的删除权限。

（3）对（1）和（2）创建的用户执行相应的操作进行权限验证。

思考题

（1）SQL Server 采用几级安全验证？简述其安全模式？

（2）简述服务器角色和数据库角色的作用？