云南大学数学与统计学院
实验报告

课程名称：数据库系统原理实验	**学期**：2017~2018 学年 第一学期	**成绩**：
指导教师：赵越	**学生姓名**：刘鹏	**学生学号**：20151910042
实验名称：SQL Server 的上机环境及语言元素		
上机实践编号：No.01	**实验日期**：第 1-2 周	**实验学时**：2 学时
学院：数学与统计学院	**专业**：信息与计算科学	**年级**：2015 级

一、 实验目的

1. 学习并掌握 Microsoft SQL Server 的上机环境；
2. 熟悉 Microsoft SQL Server 的语言元素；
3. 熟悉自由表的创建。

二、实验内容

配合教材[1]第一、二章的内容，要求：

1. 熟悉 SQL server 的上机环境：

   - 熟悉系统界面，
   - 学会使用对象资源管理器
   - 学会建立数据库、建立表
   - 学会使用 Editor 窗口。

2. 表的设计与建立

   - 练习用图形化方法建立表；
   - 熟悉对表中数据的添加、浏览、查看、维护。

3. 熟悉并了解 SOL Server 的数据及数据元素：

   - 常量、变量及其数据类型
   - 常用函数
   - 表达式与运算

三、实验平台

Windows 10 1709 Enterprise 中文版；
Microsoft Visual Studio 2017 Enterprise
Microsoft SQL Server 2017 for Developers
Microsoft SQL Server Management Studio 14.0.17199.0

四、实验记录与实验结果分析

1. 熟悉 SQL Server Management Studio 的界面[2]

启动 SQL Server Management Studio，熟悉其用户界面，打开 SSMS 如下图所示：

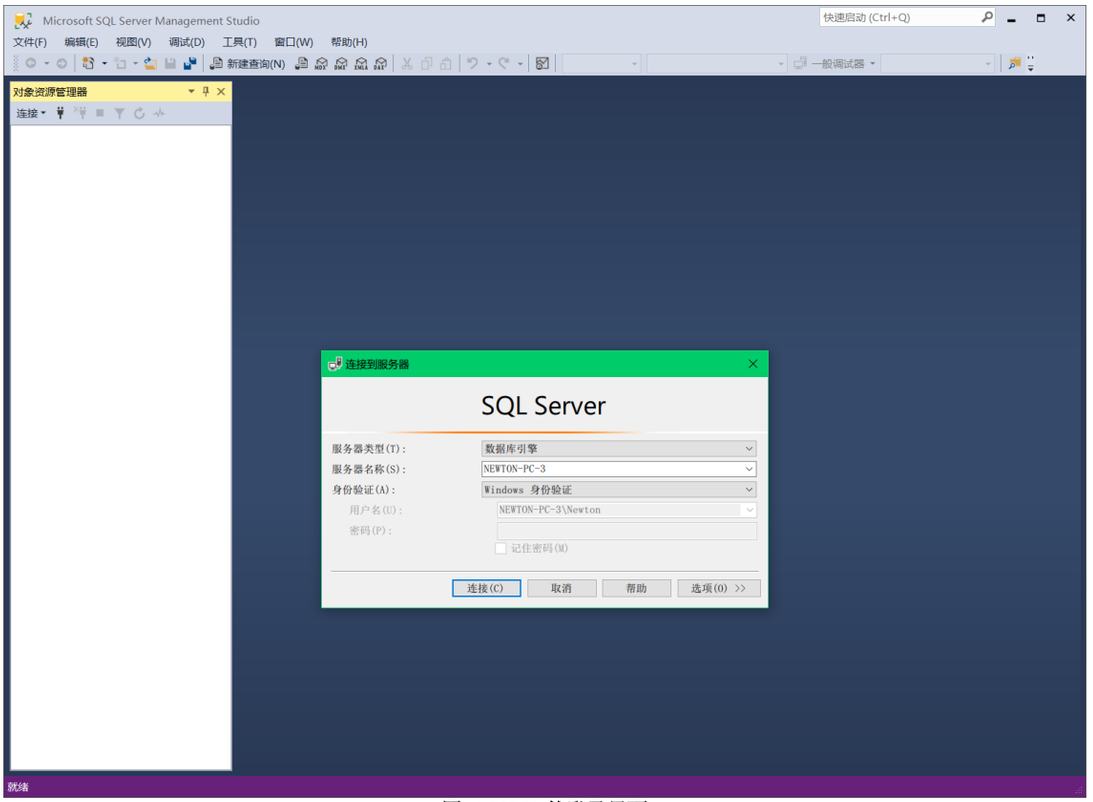

图 1 SSMS 的登录界面

从管理角度来看，一个数据库的改动以及后期维护，都要有一定的权限[2]，而与权限绑定的是操作者的身份，可以看到，在连接窗口中，有服务器类型、服务器名称、身份验证三个项目，这就是为了限制无关人等访问数据库的手段。

打开数据库，完成连接：

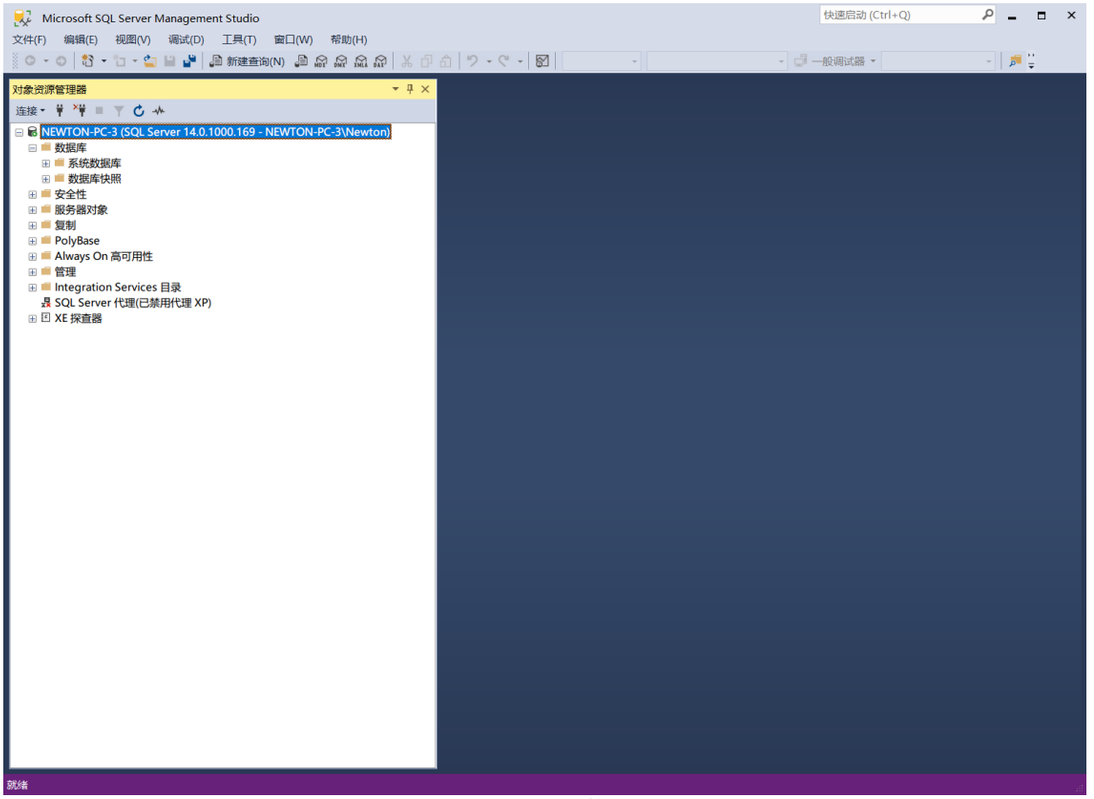

图 2 打开 SSMS 之后的界面

2. 通过图形化的方式创建一个数据库

创建一个数据库，调整数据库的位置，初始化数据库的相关配置：

这里要特别注意，因为在建库这个很严肃的过程发生之前，首先要严格规划一些参数，毕竟数据库是存放数据的地方，而数据又是财产的具体化。所以需要将数据保存到安全的磁盘上，那种有可能会坏掉的硬盘就不适合存放数据库，没有加 RAID 卡的小磁盘，也不适合存放有可能会扩展得很大的数据库。所以，经过我的考虑，决定将这个新建的数据库，放到我的磁盘 D 中，这是一个 intel 出品的 NVMe 固态硬盘，带有完整的掉电保护。具体操作如下：

先是建库，所有的相关操作，都在 UI 的右边的对象资源管理器里面，这里有数据库、安全性、服务器对象、复制、PolyBase、Always On 高可用性、管理、Integration Services 目录、SQL Server 代理、XE 探查器。

由于微软设计师想让 SQL Server 运行在装有 Windows Server 操作系统的电脑上，所以与当前使用的系统不是非常匹配，但是 Windows Server 2017 与 Windows 10 Enterprise 在 UI 上同属 Windows 10 系列，所以匹配度也算可以，足以满足一般练习性要求。

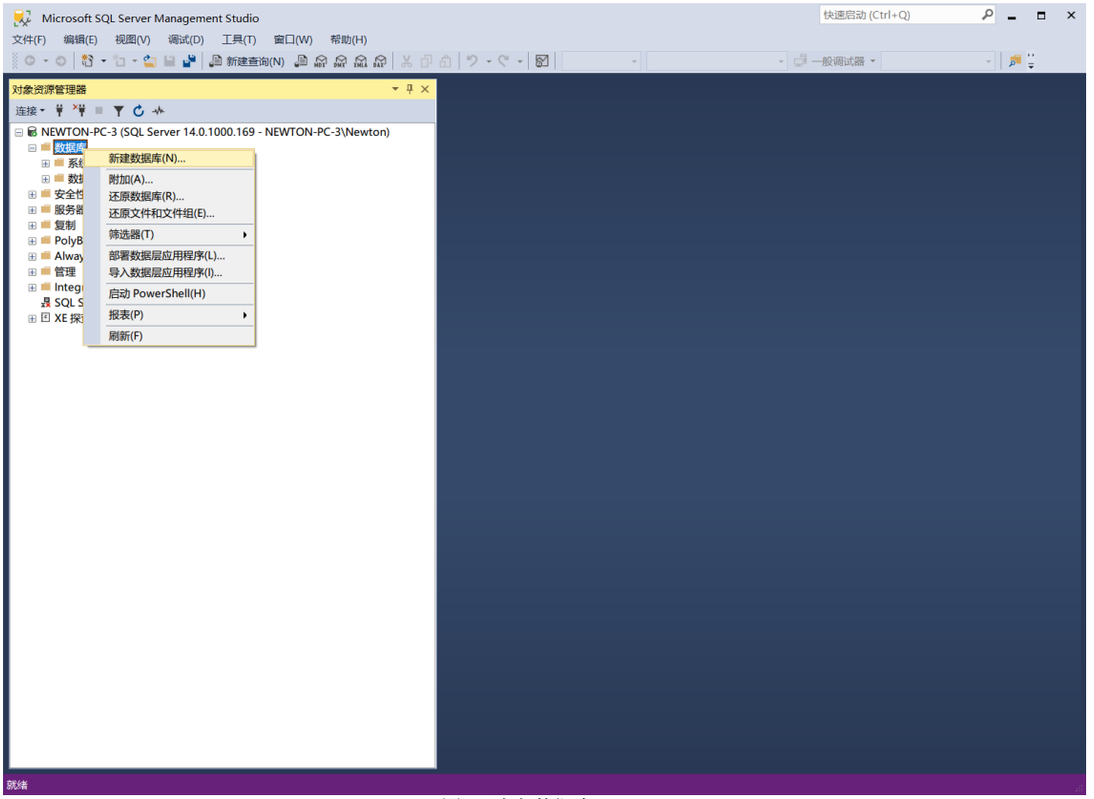

图 3 建立数据库

然后是选择初始值：

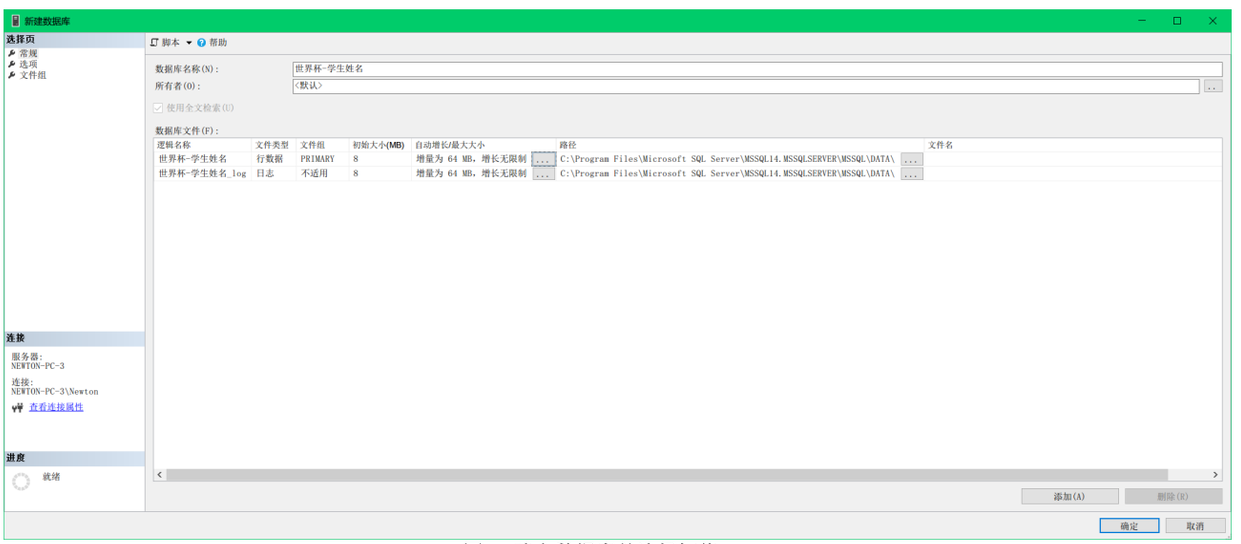

图 4 建立数据库并选择初值

建成之后，数据资源管理器底下的数据库选项中，就会多出一个名为“世界杯-学生姓名”的库。通过右键菜单，检视可以对这个新出现的对象进行怎样的操作：

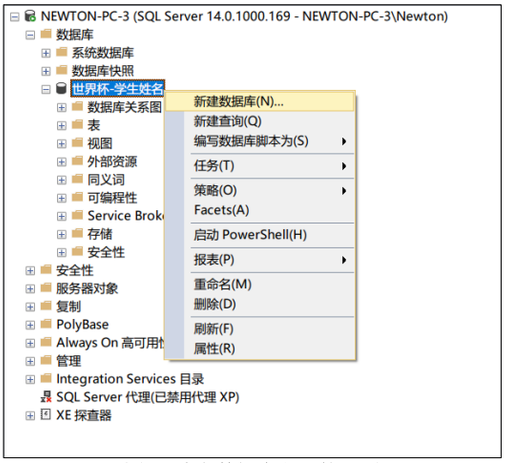

图 5 建立数据库之后的界面

3. 通过图形化方式在数据库下建立表

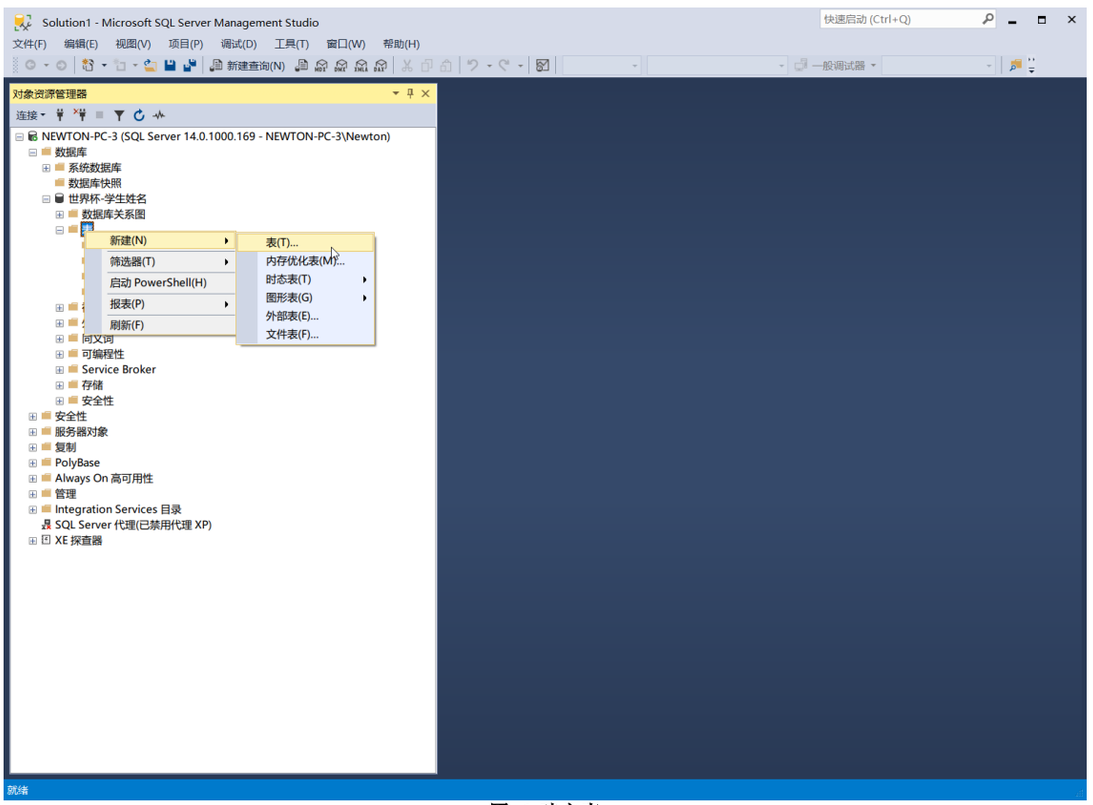

图 6 建立表

右击数据库底下的表选项，选择新建，表。在弹出的对话框中输入辅助教材中的基本信息。结果如下：

图 7 建立表之后的界面

4. SQL Server 2017 的数据类型

要了解一个数据库，必然要了解这个数据库中所有的数据类型。虽然 Visual FoxPro 的底层是一个 Access 数据库，但是它的数据类型与 Access 还是有很大不同的。至于在这里用到的 SQL Server，那就更不一样了。详情参考微软的官方手册。大致分来，有：

Character 字符串

表格 1 character 类型

数据类型	描述	存储
char(n)	固定长度的字符串。最多 8, 000 个字符。	n
varchar(n)	可变长度的字符串。最多 8, 000 个字符。	
varchar(max)	可变长度的字符串。最多 1, 073, 741, 824 个字符	
text	可变长度的字符串。最多 2GB 字符数据。	

Unicode 字符串

表格 2 Unicode 类型

数据类型	描述	存储
nchar(n)	固定长度的 Unicode 数据。最多 4, 000 个字符	
nvarchar(n)	可变长度的 Unicode 数据。最多 4, 000 个字符	
nvarchar(max)	可变长度的 Unicode 数据。最多 536, 870, 912 个字符	
ntext	可变长度的 Unicode 数据。最多 2GB 字符数据。	

Binary 类型

表格 3 binary 类型

数据类型	描述	存储
bit	允许 0、1 或 NULL	
binary(n)	固定长度的二进制数据。最多 8, 000 字节	
varbinary(n)	可变长度的二进制数据。最多 8, 000 字节	
varbinary(max)	可变长度的二进制数据。最多 2GB 字节	
image	可变长度的二进制数据。最多 2GB。	

Number 类型

表格 4 number 类型

数据类型	描述	存储
tinyint	允许 0 到 255 的所有数字	
smallint	允许-32, 768 到 32, 767 的所有数字	
int	允许从-2,147, 483, 648 到 2, 147, 483, 647	
bigint	允许介于-9 223 372 036 854 775 808 和 9 223 372 036 854 775 807 之间所有的数字。	
decimal(p, s)	固定精度和比例的数字，允许从 $-10^{38}+1$ 到 $10^{38}+1$ 之间的数字。 p 参数指示可以存储最大位数（小数点左侧和右侧）。P 必须是 1 到 38 之间的值。默认是 18。 s 参数指示小数点右侧存储的最大位数。S 必须是 0 到 p 之间的值。默认是 0。	
numeric(p, s)	固定精度和比例的数字。允许从 -10^38 +1 到 10^38 -1 之间的数字。 p 参数指示可以存储的最大位数（小数点左侧和右侧）。p 必须是 1 到 38 之间的值。默认是 18。 s 参数指示小数点右侧存储的最大位数。s 必须是 0 到 p 之间的值。默认是 0。	
smallmoney	介于 -214,748.3648 和 214,748.3647 之间的货币数据。	
money	介于 -922,337,203,685,477.5808 和 922,337,203,685,477.5807 之间的货币数据。	
float(n)	从 -1.79E + 308 到 1.79E + 308 的浮动精度数字数据。参数 n 指示该字段保存 4 字节还是 8 字节。float(24) 保存 4 字节，而 float(53) 保存 8 字节。n 的默认值是 53。	
real	从 -3.40E + 38 到 3.40E + 38 的浮动精度数字数据。	

Date 类型

表格 5 date 类型

数据类型	描述	存储
datetime	从 1753 年 1 月 1 日到 9999 年 12 月 31 日，精度为 3.33 毫秒。	
datetime2	从 1753 年 1 月 1 日到 9999 年 12 月 31 日，精度为 100 纳秒。	
smalldatetime	从 1900 年 1 月 1 日到 2079 年 6 月 6 日，精度为 1 分钟。	
date	仅存储日期。从 0001 年 1 月 1 日到 9999 年 12 月 31 日。	
time	仅存储时间。精度为 100 纳秒。	
datetimeoffset	与 datetime2 相同，外加时区偏移	
timestamp	存储唯一的数字，每当创建或修改某行时，该数字会更新。timestamp 基于内部时钟，不对应真实时间。每个表只能有一个 timestamp 变量。	

其他数据类型

表格 6 其他数据类型

数据类型	描述	存储
sql_variant	存储最多 8,000 字节不同数据类型的数据，除了 text、ntext 以及 timestamp。	
uniqueidentifier	存储全局标识符 (GUID)。	
xml	存储 XML 格式化数据。最多 2GB，	
cursor	存储对用于数据库操作的指针的引用。	
table	存储结果集，供稍后处理。	

在新建的表中，创建需要的列字段，然后保存即可。这时候点击刷新，就会出现刚刚建立的这个表。在 Visual FoxPro 中，表的格式是 dbf，但是在 SQL Server 中，是一个“dbo.基本信息”的文件，这是 SQL Server 的默认模式。首要的是列，展开列，就会看到输入的属性。要想编辑这个表，就右击 **dbo.基本信息**，进行图形化编辑，可以在对象资源管理器里修改列名称，可以在工作窗口里面修改每一个元组的值。

可以通过图形化方式将所有的属性都改成英文的，包括这个表本身。

图 8

5. 通过 SQL 脚本语句对数据库进行操作[3]

详见六、T-SQL 的归纳，其中的例子均是基于真实数据进行编写。

五、实验思考

1. 为什么要使用 SQL Server 对象资源管理器

使用图形界面，集中连接、维护、修改与创建数据库，比较简便。可以在一个侧边窗口里面看到所有的数据库，十分简洁。

2. SQL Server 建立表的步骤及操作

方法 1：在对象资源管理器里，在数据库名字上单击右键，插入表，就可以生成一个表，同时打开窗口，对表的属性列进行编辑。这种方法比较快捷。

方法 2：在 sqlcmd 里面，进行设置。直接连接到服务器，比如在我的实验里，用的是本机的服务，所以在 PowerShell 里面，直接输入，然后可以如下操作：

```
PS C:\Users\Newton> SQLCMD.EXE -S newton-pc-3
1> use 信息与计算科学 2015 级学生信息
2> go
已将数据库上下文更改为 "信息与计算科学 2015 级学生信息"。
1> create table [dbo].Test
2> (row1 char(10) not null,
3>  row2 char(10) --name
4> );
5> go
1> insert into [dbo].Test
2> values('20151110','Newton')
3> go

(1 行受影响)
1> insert into [dbo].Test
2> values('20121111','Lagrange')
3> go

(1 行受影响)
1> select * from [dbo].Test
2> order by row1
3> go
row1       row2
---------- ----------
20121111   Lagrange
20151110   Newton

(2 行受影响)
1>
```

程序示例 1

可以在这个例子中看到，通过命令行的形式，在数据库“信息与计算科学 2015 级学生信息”中建立了一个表，同时定义了相应的列，而且插入了两个值。同时可以执行查询语句。在控制台输入一行行的命令，然后最后输入一个 go，回车之后就可以执行从上面的 1>到末尾的所有语句，如果有错误，会指出错误所在的行。

这种命令交互，与 Python 相对而言有点区别，Python 的 console 是摁下回车一次，就执行一次。（当然，冒号后的语句除外。）

3. 对命令窗口的使用进行总结

（如何测试函数、表达式；如何使用命令、编辑命令等）；

```
PS C:\Users\Newton> SQLCMD.EXE -S newton-pc-3
1> use 信息与计算科学 2015 级学生信息
2> go
已将数据库上下文更改为 "信息与计算科学 2015 级学生信息"。
1> select 姓名 from dbo.个人基本信息
2> where 姓名 like "刘%"
3> order by 姓名
4> go
姓名
----------------------------------------------------
刘航麟
刘鹏
刘苏文
(3 行受影响)
1>
```

程序示例 2

测试函数与表达式，直接在命令行里面输入即可。

4. 对 SQL Server 的常用数据类型进行总结

详见四、实验记录与实验结果分析中的相关表格。

六、T-SQL 的归纳

1. T-SQL 的语句、变量与符号

1.1 T-SQL 语句概述

Transact-SQL 语句都是由一个谓词开始，该谓词描述这条语句要产生的动作，如 select 或 update 关键字。谓词后紧接着一个或多个子句（Clause），子句中给出了被谓词作用的数据或提供谓词动作的详细信息，每一条子句都由一个关键字开始。

```
select 子句
[into 子句]
[from 子句]
[where 子句]
[group by 子句]
[having 子句]
[order by 子句]
```

Transact-SQL 语言主要由以下几部分组成。

（1） 数据定义语言（Data Definition Language, DDL）：用于在数据库系统中对数据库、表、视图、索引等数据库对象进行创建和管理

（2） 数据控制语言（Data Control Language, DCL）：用于实现对数据库中数据的完整性、安全性等的控制。

（3） 数据操纵语言（Data Manipulation Language, DML）：用于插入、修改、删除和查询数据库中的数据。

Transact-SQL 语句的分类如下

（1） 变量说明语句：用来说明变量的命令。

（2） 数据定义语句：用来建立数据库、数据库对象和定义列，大部分是以 create 开头的命令，如 create table、create view 和 drop table 等。

（3） 数据操纵语句：用来操纵数据库中数据的命令，如 select、insert、update、delete 和 cursor 等。

（4） 数据控制语句：用来控制数据库组件的存取许可、存取权限等命令，如 grant、revoke 等。

（5） 流程控制语句：用于设计应用程序流程的语句，如 if while 和 case 等。

（6） 内建函数：说明变量的命令。

（7） 其他命令：嵌入进命令中进行使用的标准函数。

1.2 T-SQL 的常量与变量

T-SQL 中有数字常量、字符串常量，这两者与其他语言一般没有区别，可以借鉴使用；T-SQL 还有时间和日期常量，其格式因国家不同而有差别。T-SQL 有几个保留字，代表了一些常用的数据值，如：

```
current_time
current_timestamp
```

局部变量：用户可以自定义的变量，其作用范围仅在程序内部。局部变量的名称是用户自定义的，命名的局部变量名要符合 SQL Server 的标识符命名规则，且局部变量名必须以“@”开头。、

局部变量的声明需要使用 declare 语句，其格式如下：

```
use database_name
declare @LocalVariable char(10)
select @localVariable = Sno from tb_student where Sdep = '会计学'
print @localVariable
```

值得注意的是，select 语句在这里兼有赋值的作用，而不仅仅是查询。如果是单纯的赋值，需要用下面的语句

```
declare @LocalVariable int
set @LocalVariable = 10
```

程序示例 3

全局变量是 SQL Server 系统内部事先定义好的变量，不用用户参与定义，对用户而言，其作用范围并不局限于某一应用程序，而是任何程序均可以随时调用。全局变量通常用于存储一些 SQL Server 的配置设定值和效能统计数据。

SQL Server 一共提供了 30 多个全局变量。全局变量的名称都是以@@开头的。

1.3 T-SQL 的符号

注释符：不是可执行语句，不参与程序的编译，通常是一些说明性的文字，对代码的功能或者代码的实现方式给出简要的解释和提示。--，用于单行注释，这也是 ANSI 标准的注释符。/* */，可以进行多行注释。在 SSMS 中，选中一个代码块，Ctrl Shift C，可以将其注释掉，再按下 Ctrl Shift R，取消注释。

赋值运算符：同上 select 语句与 set 语句

比较运算符：同 C 语言，但是!=（不等于），!>（不大于），!<（不小于）都不是 ANSI 的标准运算符，<>是标准的不等于，其余两个同标准 C。比较运算符的结果有 true，false，unknown 三种。

逻辑运算符：

表格 7

运算符	行为
all	如果一个比较集中全部都是 true，则值为 true
and	如果两个布尔表达式都是 true，则值为 true
any	如果一个比较集中有一个或几个为 true，则值为 true，同 some
between	如果操作数是在某个范围内，则值为 true
exists	如果子查询包含任何行，则值为 true
in	如果操作数与一个表达式列表中的某个相等，则值为 true
like	如果操作数匹配某个模式，则值为 true
not	对任何其他的布尔运算符取反
or	如果任何一个布尔表达式似乎 true，则值为 true
some	如果一个比较集中的某些为 true，则值为 true

如下实例

```
use db_student
select * from tb_student
where stu_sex = '女' and stu_age > 24
```

程序示例 4

SQL Server 还支持**位运算符**。目前还不太常用，暂不介绍。

连接运算符：加号“+”，用于连接两个或者两个以上的字符串或者二进制串、列名或者串和列的混合体，将一个串加入到另一个串的末尾，这一点类似于 Python 中的字符串加法。例如：

```
declare @name char(20)
set @name = '最爱'
print '电影名称：' + @name
```

程序示例 5

通配符：匹配指定范围内或者属于方括号所指定的集合中的任意单个字符。可以在设计模式匹配的字符串比较中使用这些通配符。

表格 8

通配符	描述	实例
%	包含零个或更多字符的任意字符	"loving%"可以表示："loving"、"loving you"、"loving anyone"
_（下划线）	任意单个字符	"loving_"可以表示："lovingc"。后面只能再接一个字符
[]	指定范围（[a~f]）或集合（[abcdef]）中的任意单个字符	[0~9]123 表示以 0~9 之间任意一个字符开头，以"123"结尾的字符
[^]	不属于指定范围（[a~f]）或集合（[abcdef]）中的任何单个字符	[^0~5]123 表示不以 0~5 之间任意一个字符开头，却以"123"结尾的字符

2. T-SQL 的流程控制语句

表格 9

begin…end	waitfor	goto
while	if…else	break
return	continue	

2.1 begin…end

begin…end 语句用于将多个 Transact-SQL 语句组合为一个逻辑块。当流程控制语句必须执行一个包含两条或者两条以上的 T-SQL 语句的语句块时，使用 begin…end 语句。此语句主要用来和其他语句结合使用。

如用于交换两个局部变量的值：

```
1>  declare @x int, @y int, @t int
2>  set @x = 1
3>  set @y = 2
4>  begin
5>  set @t = @x
6>  set @x = @y
7>  set @y = @t
8>  end
9>  print @x
10> print @y
11> go
 2
 1
1>
```

程序示例 6

2.2 if 与 if…else

if 语句可以控制程序的执行方向。

```
1> declare @x int
2> set @x = 3
3> if @x > 0
4> print '@x 是正数'
5> print 'end'
6> go
@x 是正数
end
1>
```

程序示例 7

if…else 语句示例如下，这个语句的用法和 C 语言基本一样。

```
1>  declare @x int, @y int
2>  set @x = 8
3>  set @y = 3
4>  if @x > @y
5>  print '@x > @y'
6>  else
7>  print '@x <= y'
8>  go
@x > @y
1>
```

程序示例 8

当然，这里的 if…else 可以嵌套。

2.3 case

使用 case 语句可以方便地实现多重选择的情况。case 语句的应用分为两种情况，第一种是：

```
case input_expression
    when when_expression then result_expression
    ...
    else else_result_expression
end
```

第二种是用 case 进行搜索：

```
case
    when boolean_expression then result_expression
    ...
    else else_result_expression
end
```

当第一种情况下，可以做如下例子：

```
PS C:\Users\Newton> SQLCMD.EXE -S newton-pc-3
1>  use 信息与计算科学 2015 级学生信息
2>  go
已将数据库上下文更改为 "信息与计算科学 2015 级学生信息"。
1>
2>  alter table [dbo].学期 1
3>  add 姓氏备注 char(10)
4>  go
1>
2>  update [dbo].学期 1
3>  set [姓氏备注] = case substring([姓名], 1, 1)
4>  when "刘" then "姓刘"
5>  when "郑" then "姓郑"
6>  when "游" then "姓游"
7>  when "张" then "姓张"
8>  else ""
9>  end
10>  from [dbo].学期 1
11>  go

(47 行受影响)
```

```
1>
2> select 学号, 姓名, 姓氏备注 from [dbo].学期1
3> where 姓氏备注 != ""
4> go
学号           姓名       姓氏备注
---------------------------------
20151910007    张锡成     姓张
20151910014    张建宁     姓张
20151910028    刘苏文     姓刘
20151910029    郑茂森     姓郑
20151910042    刘鹏       姓刘
20151910055    张浩       姓张
20151910115    张贺源     姓张
20151910134    刘航麟     姓刘

(8 行受影响)
1>
```

程序示例 9

当是第 2 种情况时，可以测试一下效果，如下：

```
1> update [dbo].学期1
2> set 数学分析水平 = case
3> when [数学分析(1)] >=90 then"成绩优秀"
4> when [数学分析(1)] < 90 and [数学分析(1)] >= 80 then"成绩良好"
5> when [数学分析(1)] < 80 and [数学分析(1)] >= 70 then"成绩及格"
6> else "不及格"
7> end
8> from [dbo].学期1
9> go

(47 行受影响)
1> select [姓名], [数学分析(1)] from [dbo].学期1
2> where [数学分析(1)] >= 90
3> go
姓名       数学分析(1)
-----------------------
唐效禹     90.0
张浩       90.0
(2 行受影响)
1> select [姓名],[数学分析水平] from [dbo].学期1
2> where [数学分析水平] like "%优秀"
3> go
姓名       数学分析水平
-----------------------
唐效禹     成绩优秀
张浩       成绩优秀
(2 行受影响)
1>
```

程序示例 10

在原始成绩单的基础上新增加一列，叫做“数学分析水平”，然后跟据每个人的数学分析成绩进行评判，这里需要用到 case 搜索函数。

2.4 while

作为循环的一个语句，基本用法如下：

```
while while_expression
begin
    main_body
end
```

2.5 while…continue…break

具体用法同 C 语言，这里不再赘述。

2.6 return

return 语句用于从查询或者过程中无条件退出。return 语句可以在任何时候用于从过程、批处理或者语句块中退出，位于 return 语句之后的语句不会被执行。

如果程序出错，系统会根据结果返回一个内定值，根据这个内定值，可以判断出了什么问题。一般返回一个负数，如果返回 F，说明正常执行成功。返回负数，说明有问题，负数的绝对值越大，说明问题越严重。

2.7 goto

这个语句与 C 语言中的相似，但是这里并没有 flag 语句，而是与 Python 类似，在标记符号之后加一个冒号，之后写语句，goto 语句之后直接写标记符号。这样一来就可以指定流程的方向。

2.8 waitfor

waitfor 指定触发器、存储过程或事务执行的时间、时间间隔或时间；还可以用来暂停程序的执行，指导所设定的等待时间已经过去才继续往下执行。语法格式如下：

```
waitfor delay <时间> | time <时间>
```

其中，时间必须是 datetime 类型，但不能包括日期，小时-分钟-秒就是标准定义。

- 当是 delay 时，时间指的是等待的时间长度；
- 当是 time 时，时间指的是等待结束的时刻。

七、实验体会

数据库管理软件，是建立在一个数据库系统软件之上，令数据库的管理者与数据库进行交互的工具。

之前在学习中，我一直陷入了一个误区，那就是认为 SQL 这种语言与 C/C++、Python 之类的语言一样，是可以重复执行，之后调试结果的。更可怕的是，我认为 SQL 语言才是最值得学习的。这种认识在现在的我看来，非常愚蠢。设计数据库与编写一些练习性的程序是完全不同的，设计、开发、应用数据库是一个成体系化、工程化的事情，有的时候弄错了一个节点，就需要删掉整个库，然后重新做一遍。这也就是说，建库的初期数据积累非常重要，库的初期设计更加重要。

当然，可以在命令行里面不停试错，但是执行成功就一般不能重复做了，除非是纯查询语句。但是 SQL Server 的窗口报错比较简陋，没有任何相关提示，只会说在哪里有错，错误类型与可能的错误原因都不给出。

通过对 SQL 语言的理解，我也掌握了一些数据库的基本知识。

八、参考文献

[1] 梁洁. Visual FoxProb 程序设计基础[M]. 北京: 高等教育出版社; 2010.
[2] 明日科技. SQL Server 从入门到精通[M]. 北京: 清华大学出版社; 2012.
[3] 王珊, 萨师煊. 数据库系统概论[M]. 北京: 高等教育出版社; 2014.9.