

经典 SQL. 2000 教程

实验二 数据表创建和维护

2.1 数据表管理

在使用数据库的过程中，接触最多的就是数据库中的表。表是数据存储的地方，是数据库中最重要的一部分。管理好表也就管理好了数据库。

2.1.1 数据库表的列定义

表是由行和列组成的。创建表的过程主要就是定义表的列的过程，为此，应先了解表的列的属性和数据类型。

一、列的属性

表的列名在同一个表中具有惟一性，同一列的数据属于同一种数据类型。除了用列名和数据类型来指定列的属性外，还可以定义其它属性：是否为空、默认值、标识符列、全局唯一标识符列等。

A. 是否为空

 帮助索引：<空值>

如果表的某一列被指定具有 NULL 属性，那么就允许在插入数据时省略该列的值。反之，如果表的某一列被指定具有 NOT NULL 属性，那么就不允许在没有指定列缺省值的情况下插入省略该列值的数据行。

B. 默认值

 帮助索引：<默认约束>

每当在表中为该列插入带空值的行时，显示该列的默认值。

C. 标识符列 (IDENTITY)

 帮助索引：<标识符列>

每当向具有标识符列的表中插入值时，系统会自动生成一个标识值，其值是通过递增种子值而来的。

一个表最多只能有一列定义为标识符列，而且该列必须以 `decimal`、`int`、`numeric`、`smallint`、`bigint` 或 `tinyint` 数据类型定义。在定义标识符列时，还必须指定种子和增量值（其默认值均为 1）。标识符列不允许空值，也不能包含默认值。

D. 全局唯一标识符列(RowGuid)

 帮助索引: <全局唯一标识符>

尽管标识符列属性自动为表生成行号,但不同表的标识符列可以生成相同的行号。这是因为标识符列属性只须在所使用的表上保持唯一。如果应用程序需要生成在整个数据库或世界各地所有网络计算机的全部数据库中均为唯一的标识符列,使用全局唯一标识符列属性。

一个表最多只能有一列定义为是 RowGuid 列,且该列必须定义为 `uniqueidentifier` 数据类型。

二、列数据类型

数据类型的分类

整数数据类型: `INT`、`INTEGER`、`SMALLINT`、`TINYINT`、`BIGINT`

浮点数据类型: `REAL`、`FLOAT`、`DECIMAL`、`NUMERIC`

二进制数据类型: `BINARY`、`VARBINARY`

逻辑数据类型: `BIT`

字符数据类型: `CHAR`、`NCHAR`、`VARCHAR`、`NVARCHAR`

文本和图形数据类型: `TEXT`、`NTEXT`、`IMAGE`

日期和时间数据类型: `DATETIME`、`SMALLDATETIME`

货币数据类型: `MONEY`、`SMALLMONEY`

特定数据类型: `TIMESTAMP`、`UNIQUEIDENTIFIER`

用户自定义数据类型: `SYSNAME`

新数据类型: `SQL_VARIANT`、`TABLE`

其中 `BIGINT`、`SQL_VARIANT`、`TABLE` 是 SQL Server 2000 中新增加的 3 种数据类型,下面分类讲述各种数据类型。

A. 整数数据类型

 帮助索引: <数据类型-SQL Server|numeric >

· `INT`、`INTEGER`

`INT`或`INTEGER`数据类型存储从 -2^{31} 到 $2^{31}-1$ 之间的所有正负整数,每个`INT`类型的数
据按 4 个字节存储,其中 1 位表示整数值的正负号,其它 31 位表示整数值的长度和大小。

· SMALLINT

SMALLINT数据类型存储从 -2^{15} 到 $2^{15}-1$ 之间的所有正负整数，每个 **SMALLINT** 类型的数据占用 2 个字节的存储空间，其中 1 位表示整数值的正负号，其它 15 位表示整数值的长度和大小。

· TINYINT

TINYINT 数据类型存储从 0 到 255 之间的所有正整数，每个 **TINYINT** 类型的数据占用 1 个字节的存储空间。

· BIGINT

BIGINT数据类型存储从 $-2^{63}(-9223372036854775807)$ 到 $2^{63}-1(9223372036854775807)$ 之间的所有正负整数，每个 **BIGINT** 类型的数据占用 8 个字节的存储空间。

B. 浮点数据类型



帮助索引: <数据类型-SQL Server|numeric >

浮点数据类型用于存储十进制小数。浮点数值的数据在 **SQL Server** 中采用上舍入 (**Round up**，或称为只入不舍) 方式进行存储。所谓上舍入是指，当（且仅当）要舍入的数是一个非零数时，对其保留数字部分的最低有效位上的数值加 1，并进行必要的进位。若一个数是上舍入数，其绝对值不会减少。如：对 3.14159265358979 分别进行 2 位和 12 位舍入，结果为 3.15 和 3.141592653590。

· REAL 数据类型

REAL 数据类型可精确到第 7 位小数，其范围为从 $-3.40E-38$ 到 $3.40E+38$ 。每个 **REAL** 类型的数据占用 4 个字节的存储空间。

· FLOAT

FLOAT 数据类型可精确到第 15 位小数，其范围为从 $-1.79E-308$ 到 $1.79E+308$ 。每个 **FLOAT** 类型的数据占用 8 个字节的存储空间。

FLOAT 数据类型可写为 **FLOAT([n])** 的形式。**n** 指定 **FLOAT** 数据的精度。**n** 为 1 到 15 之间的整数值。当 **n** 取 1 到 7 时，实际上是定义了一个 **REAL** 类型的数据，系统用 4 个字节存储它；当 **n** 取 8 到 15 时，系统认为其是 **FLOAT** 类型，用 8 个字节存储它。

· DECIMAL

DECIMAL数据类型可以提供小数所需要的实际存储空间，但也有一定的限制，您可以用 2 到 17 个字节来存储从 $-10^{38}-1$ 到 $10^{38}-1$ 之间的数值。可将其写为 **DECIMAL**[(p, [s])] 的形式，p和s确定了精确的比例和数位。其中p表示可供存储的值的总位数（不包括小数点），缺省值为 18；s表示小数点后的位数，缺省值为 0。例如：**decimal(15,5)**，表示共有 15 位数，其中整数 10 位，小数 5 位。下表列出了各精确度所需的字节数之间的关系。

精 度	字 节 数	精 度	字 节 数
1~2	2	20~21	10
3~4	3	22~24	11
5~7	4	25~26	12
8~9	5	27~28	13
10~12	6	29~31	14
13~14	7	32~33	15
15~16	8	34~36	16
17~19	9	37~38	17

· NUMERIC

NUMERIC 数据类型与 **DECIMAL** 数据类型完全相同。

C. 二进制数据类型

 帮助索引：<数据类型-SQL Server|binary >

· BINARY

BINARY 数据类型用于存储二进制数据。其定义形式为 **BINARY**(n)，n 表示数据的长度，取值为 1 到 8000。在使用时必须指定 **BINARY** 类型数据的大小，至少应为 1 个字节。**BINARY** 类型数据占用 n+4 个字节的存储空间。在输入数据时必须在数据前加上字符“0X”作为二进制标识，如：要输入“abc”，则应输入“0xabc”。若输入的数据过长，将会截掉其超出部分。若输入的数据位数为奇数，则会在起始符号“0X”后添加一个 0，如上述的“0xabc”会被系统自动变为“0x0abc”。

· VARBINARY

VARBINARY 数据类型的定义形式为 **VARBINARY**(n)。它与 **BINARY** 类型相似，n 的取值也为 1 到 8000，若输入的数据过长，将会截掉其超出部分。不同的是 **VARBINARY** 数据类

型具有变动长度的特性，因为 **VARBINARY** 数据类型的存储长度为实际数值长度+4 个字节。

当 **BINARY** 数据类型允许 **NULL** 值时，将被视为 **VARBINARY** 数据类型。

一般情况下，由于 **BINARY** 数据类型长度固定，因此它比 **VARBINARY** 类型的处理速度快。

D. 逻辑数据类型



帮助索引: <数据类型-SQL Server|numeric >

BIT 数据类型占用 1 个字节的存储空间，其值为 0 或 1。如果输入 0 或 1 以外的值，将被视为 1。**BIT** 类型不能定义为 **NULL** 值（所谓 **NULL** 值是指空值或无意义的值）。

E. 字符数据类型



帮助索引: <数据类型-SQL Server|character >

字符数据类型是使用最多的数据类型。它可以用来存储各种字母、数字符号、特殊符号。一般情况下，使用字符类型数据时须在其前后加上单引号 ' 双引号 "。

· CHAR

CHAR 数据类型的定义形式为 **CHAR[(n)]**。以 **CHAR** 类型存储的每个字符和符号占一个字节的存储空间。**n** 表示所有字符所占的存储空间，**n** 的取值为 1 到 8000，即可容纳 8000 个 **ANSI** 字符。若不指定 **n** 值，则系统默认值为 1。若输入数据的字符数小于 **n**，则系统自动在其后添加空格来填满设定好的空间。若输入的数据过长，将会截掉其超出部分。

· NCHAR

NCHAR 数据类型的定义形式为 **NCHAR[(n)]**。它与 **CHAR** 类型相似。不同的是 **NCHAR** 数据类型 **n** 的取值为 1 到 4000。因为 **NCHAR** 类型采用 **UNICODE** 标准字符集。**UNICODE** 标准规定每个字符占用两个字节的存储空间，所以它比非 **UNICODE** 标准的数据类型多占用一倍的存储空间。使用 **UNICODE** 标准的好处是因其使用两个字节做存储单位，其一个存储单位的容纳量就大大增加了，可以将全世界的语言文字都囊括在内，在一个数据列中就可以同时出现中文、英文、法文、德文等，而不会出现编码冲突。

· VARCHAR

VARCHAR 数据类型的定义形式为 **VARCHAR[(n)]**。它与 **CHAR** 类型相似，**n** 的取值也为 1 到 8000，若输入的数据过长，将会截掉其超出部分。不同的是，**VARCHAR** 数据类型具

有变动长度的特性，因为 **VARCHAR** 数据类型的存储长度为实际数值长度，若输入数据的字符数小于 **n**，则系统不会在其后添加空格来填满设定好的空间。

一般情况下，由于 **CHAR** 数据类型长度固定，因此它比 **VARCHAR** 类型的处理速度快。

· **NVARCHAR**

NVARCHAR 数据类型的定义形式为 **NVARCHAR[(n)]**。它与 **VARCHAR** 类型相似。不同的是，**NVARCHAR** 数据类型采用 **UNICODE** 标准字符集，**n** 的取值为 **1** 到 **4000**。

F. 文本和图形数据类型



帮助索引：<数据类型-SQL Server|character >

这类数据类型用于存储大量的字符或二进制数据

· **TEXT**

TEXT数据类型用于存储大量文本数据，其容量理论上为 **1** 到 **2³¹-1** 个字节，在实际应用时需要视硬盘的存储空间而定。

SQL Server 2000 以前的版本中，数据库中一个 **TEXT** 对象存储的实际上是一个指针，它指向一个以 **8KB** 个字节为单位的数据页。这些数据页是动态增加并被逻辑链接起来的。在 **SQL Server 2000** 中，则将 **TEXT** 和 **IMAGE** 类型的数据直接存放到表的数据行中，而不是存放到不同的数据页中。这就减少了用于存储 **TEXT** 和 **IMAGE** 类型的空间，并相应减少了磁盘处理这类数据的 **I/O** 数量。

· **NTEXT**

NTEXT数据类型与**TEXT** 类型相似。不同的是，**NTEXT**类型采用**UNICODE**标准字符集，因此其理论容量为 **2³⁰-1** 个字节。

· **IMAGE**

IMAGE数据类型用于存储大量的二进制数据，其理论容量为 **2³¹-1** 个字节。其存储数据的模式与**TEXT**数据类型相同。它通常用来存储图形等**OLE**对象。在输入数据时同**BINARY**数据类型一样，必须在数据前加上字符“**0X**”作为二进制标识。

G. 日期和时间数据类型



帮助索引：<数据类型-SQL Server|datetime >

· DATETIME

DATETIME 数据类型用于存储日期和时间的结合体。它可以存储从公元 1753 年 1 月 1 日零时起到公元 9999 年 12 月 31 日 23 时 59 分 59 秒之间的所有日期和时间，其精确度可达三百分之一秒，即 3.33 毫秒。

DATETIME 数据类型所占用的存储空间为 8 个字节，其中前 4 个字节用于存储 1900 年 1 月 1 日以前或以后的天数，数值分正负，正数表示在此日期之后的日期，负数表示在此日期之前的日期。后 4 个字节用于存储从此日零时起所指定的时间经过的毫秒数。如果在输入数据时省略了时间部分，则系统将 12:00:00:000AM 作为时间缺省值；如果省略了日期部分，则系统将 1900 年 1 月 1 日作为日期缺省值。

· SMALLDATETIME

SMALLDATETIME 数据类型与 **DATETIME** 数据类型相似，但其日期时间范围较小，为从 1900 年 1 月 1 日到 2079 年 6 月 6 日；精度较低，只能精确到分钟，其分钟个位上为根据秒数四舍五入的值，即以 30 秒为界四舍五入。如：**DATETIME** 时间为 14:38:30.283 时，**SMALLDATETIME** 认为是 14:39:00。**SMALLDATETIME** 数据类型使用 4 个字节存储数据。其中前 2 个字节存储从基础日期 1900 年 1 月 1 日以来的天数，后两个字节存储此日零时起所指定的时间经过的分钟数。

下面介绍日期和时间的输入格式

日期输入格式

日期的输入格式很多，大致可分为三类：

· 英文+数字格式

此类格式中月份可用英文全名或缩写，且不区分大小写；年和月日之间可不用逗号；年份可为 4 位或 2 位，当其为两位时，若值小于 50 则视为 20xx 年，若大于或等于 50 则视为 19xx 年；若日部分省略，则视为当月的 1 号。以下格式均为正确的日期格式：

June 21 2000、Oct 1 1999、January 2000、2000 February、2000 May 1、
2000 1 Sep、99 June、July 00

· 数字+分隔符格式

允许把斜杠(/)、连接符(-)和小数点(.)作为用数字表示的年、月、日之间的分隔符。
如：

YMD: 2000/6/22、2000-6-22、2000.6.22

MDY: 3/5/2000、3-5-2000、3.5.2000

DMY: 31/12/1999、31-12-1999、31.12.2000

- 纯数字格式

纯数字格式是以连续的 4 位、6 位或 8 位数字来表示日期。如果输入的是 6 位或 8 位数字，系统将按年、月、日来识别，即 YMD 格式，并且月和日都是用两位数字来表示；如果输入的数字是 4 位数，系统认为这 4 位数代表年份，其月份和日缺省为此年度的 1 月 1 日。如：

20000601——2000 年 6 月 1 日

991212——1999 年 12 月 12 日

1998——1998 年

时间输入格式

在输入时间时必须按“小时、分钟、秒、毫秒”的顺序来输入。在其间用冒号“:”隔开。但可将毫秒部分用小数点“.”分隔，其后第一位数字代表十分之一秒，第二位数字代表百分之一秒，第三位数字代表千分之一秒。当使用 12 小时制时，用 AM(am)和 PM(pm)分别指定时间是午前或午后，若不指定，系统默认为 AM。AM 与 PM 均不区分大小写。如：

3:5:7.2pm——下午 3 时 5 分 7 秒 200 毫秒

10:23:5.123Am——上午 10 时 23 分 5 秒 123 毫秒

可以使用 SET DATEFORMAT 命令来设定系统默认的时间格式。

H. 货币数据类型



帮助索引: <数据类型-SQL Server|money >

货币数据类型用于存储货币值。在使用货币数据类型时，应在数据前加上货币符号，系统才能辨识其为哪国的货币，如果不加货币符号，则默认为“¥”。

- MONEY

MONEY数据类型的数据是一个有 4 位小数的DECIMAL值，其取值从 -2^{63}

(-922337203685477.5808)到 $2^{63}-1$ (+922337203685477.5807)，数据精度为万分之一货币单位。MONEY数据类型使用 8 个字节存储。

· SMALLMONEY

SMALLMONEY 数据类型类似于 **MONEY** 类型，但其存储的货币值范围比 **MONEY** 数据类型小，其取值从 -214748.3648 到 +214748.3647，存储空间为 4 个字节。

I. 特定数据类型

 帮助索引: <数据类型-SQL Server|timestamp >

SQL Server 中包含了一些用于数据存储的特殊数据类型。

· TIMESTAMP

TIMESTAMP 数据类型提供数据库范围内的唯一值。此类型相当于 **BINARY(8)** 或 **VARBINARY(8)**，但当它所定义的列在更新或插入数据行时，此列的值会被自动更新，一个计数值将自动地添加到此 **TIMESTAMP** 数据列中。每个数据库表中只能有一个 **TIMESTAMP** 数据列。如果建立一个名为 “**TIMESTAMP**” 的列，则该列的类型将被自动设为 **TIMESTAMP** 数据类型。

· UNIQUEIDENTIFIER

UNIQUEIDENTIFIER 数据类型存储一个 16 位的二进制数字。此数字称为 **GUID** (Globally Unique Identifier, 即全球唯一鉴别号)。此数字由 **SQL Server** 的 **NEWID()** 函数产生的全球唯一的编码，在全球各地的计算机经由此函数产生的数字不会相同。

J. 用户自定义数据类型

 帮助索引: <数据类型-SQL Server|用户定义>

SYSNAME 数据类型是系统提供给用户的，便于用户自定义数据类型。它被定义为 **NVARCHAR(128)**，即它可存储 128 个 **UNICODE** 字符或 256 个一般字符。

K. 新数据类型

 帮助索引: <数据类型-SQL Server|sql_variant, 数据类型-SQL Server|表>

SQL Server 2000 中增加了 3 种数据类型：**BIGINT**、**SQL_VARIANT** 和 **TABLE**。其中 **BIGINT** 数据类型已在整数类型中介绍，下面介绍其余两种：

· SQL_VARIANT

SQL_VARIANT 数据类型可以存储除文本、图形数据 (**TEXT**、**NTEXT**、**IMAGE**) 和

TIMESTAMP 类型数据外的其它任何合法的 SQL Server 数据。此数据类型大大方便了 SQL Server 的开发工作。

· TABLE

TABLE 数据类型用于存储对表或视图处理后的结果集。这一新类型使得变量可以存储一个表，从而使函数或过程返回查询结果更加方便、快捷。

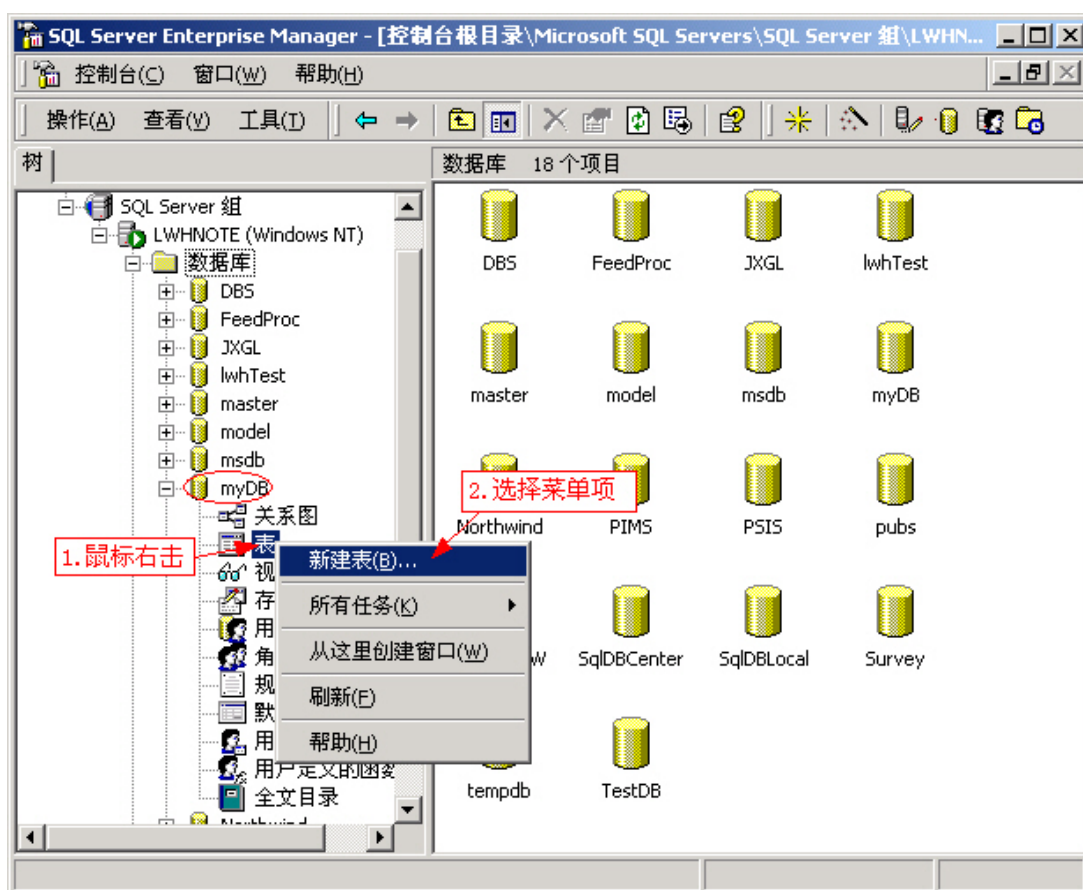
2.1.2 创建数据库表

一、用企业管理器创建数据表

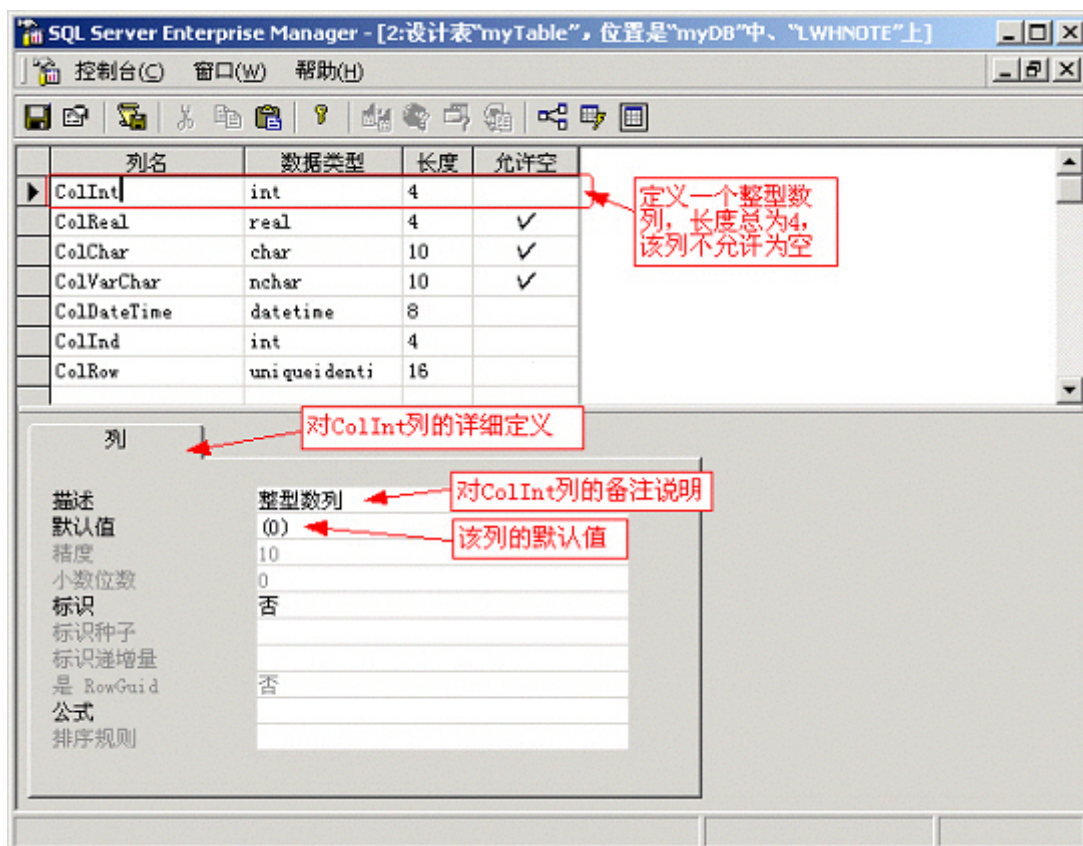
 帮助索引: <创建表>

如: 在 myDB 数据库中新建一个数据表 myTable, 演示各种数据类型列以及含有空值列、默认值列、标识符列、全局唯一标识符列属性的数据类型列。

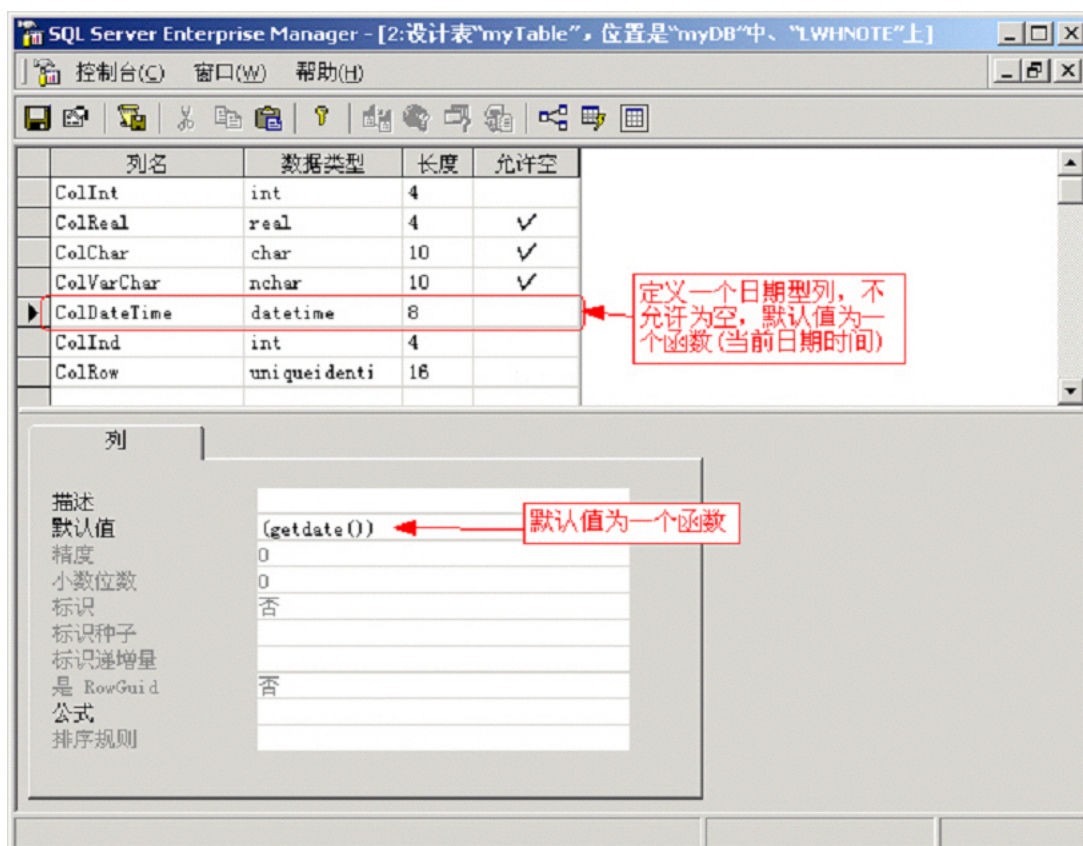
注: 如果没有myDB数据库, 参见<[1.5.1 创建数据库](#)>创建一个myDB数据库。



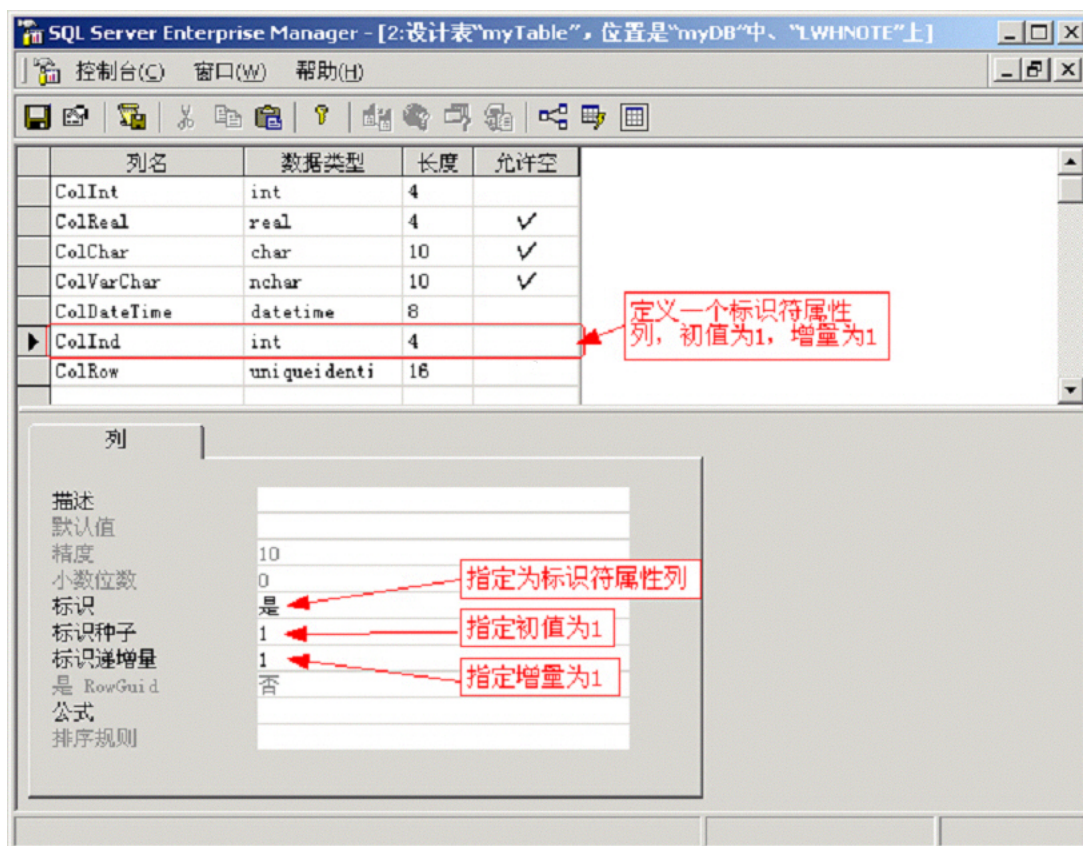
下图定义一个整型数列, 该列不能为空, 默认值为 0



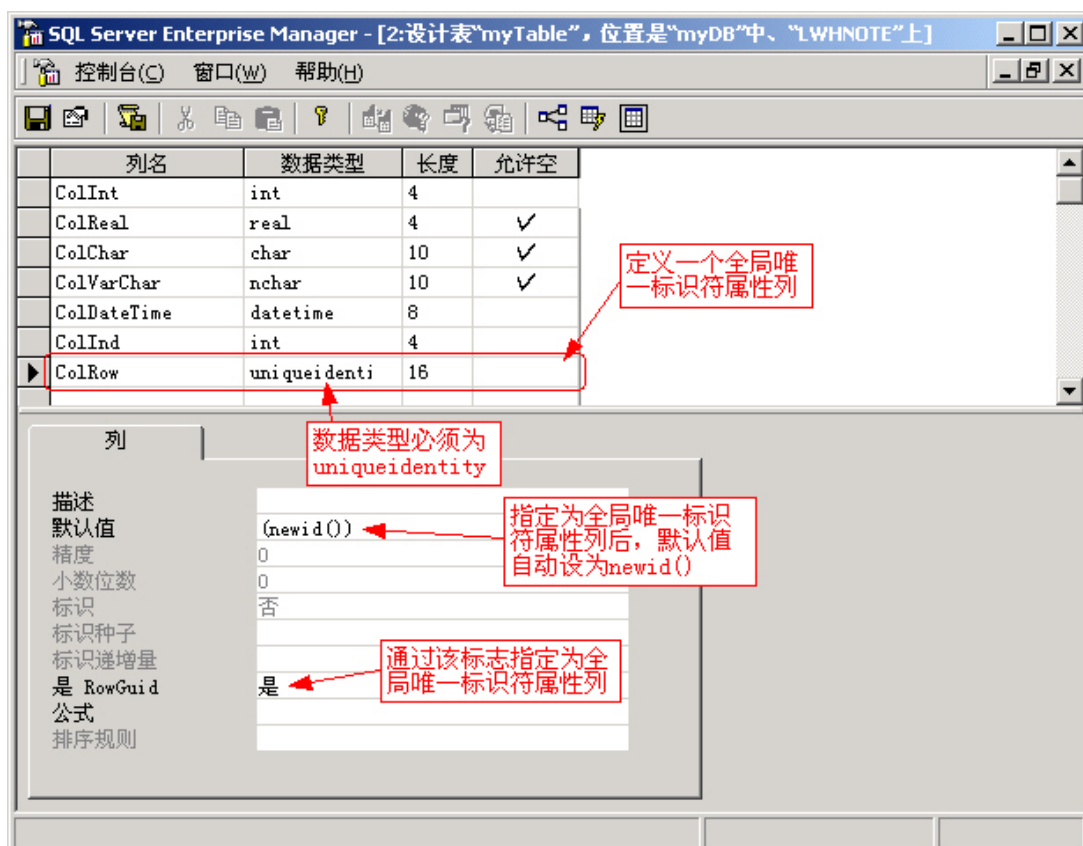
下图定义一个日期型列，该列不能为空，且缺省值为一个函数



下图定义一个标识符属性列，其初值为 1，增量也为 1



下图定义一个全局唯一标识符属性列

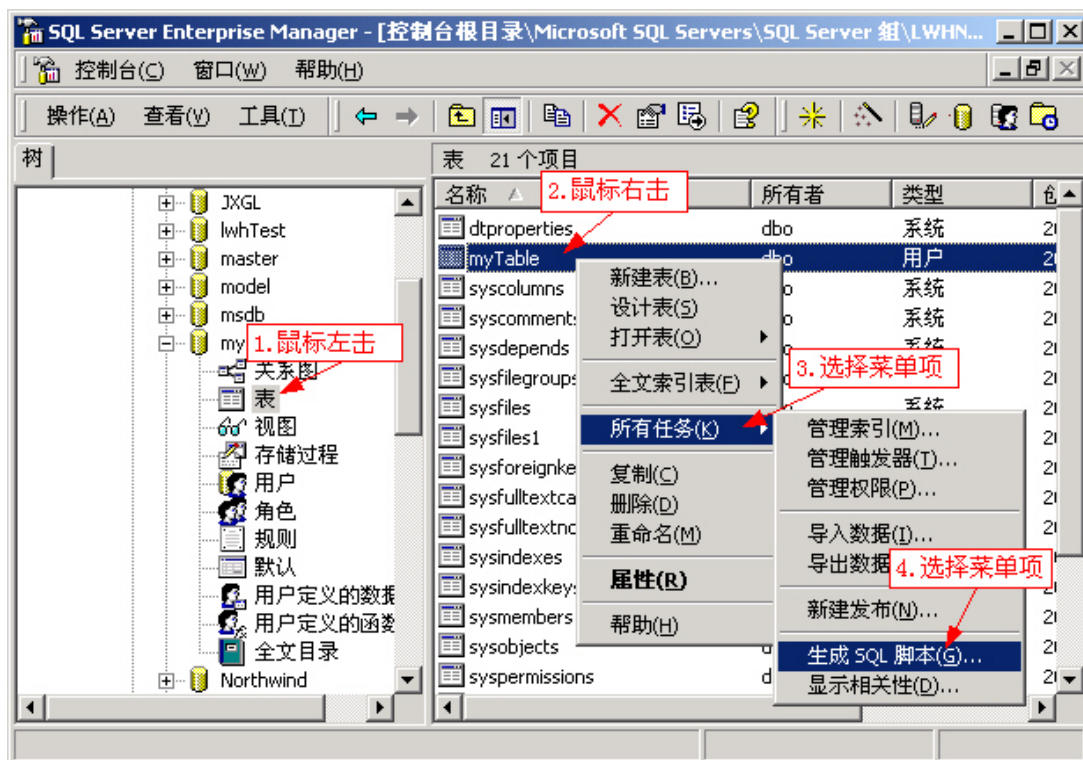


按保存按钮，由于是新建数据库表，弹出如下对话框，输入表名“myTable”，然后按“确定”，即可保存。

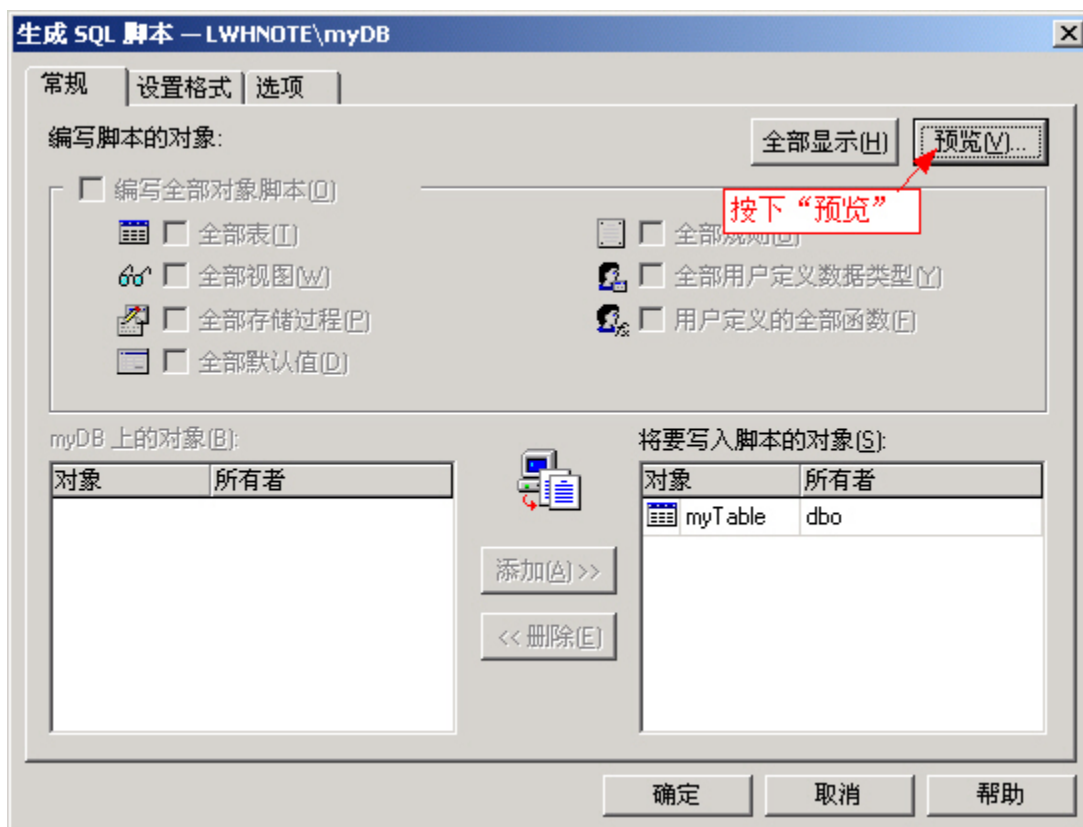


在企业管理器中，可以生成指定数据表的 Transact-SQL 语句，以便了学习

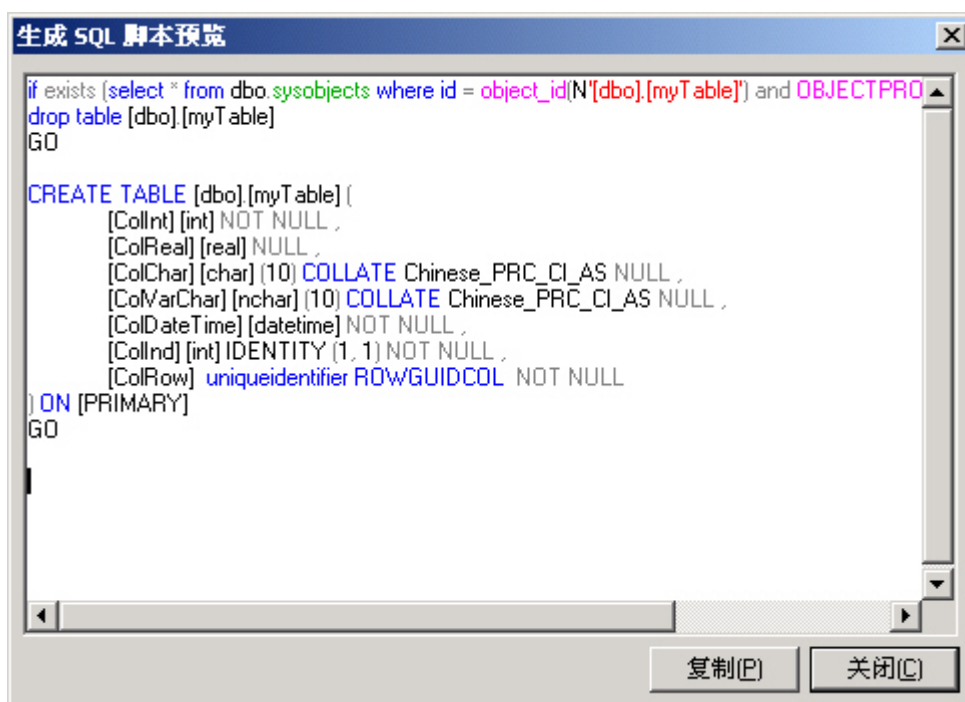
Transact-SQL 语言。



弹出如下对话框，按下“预览”



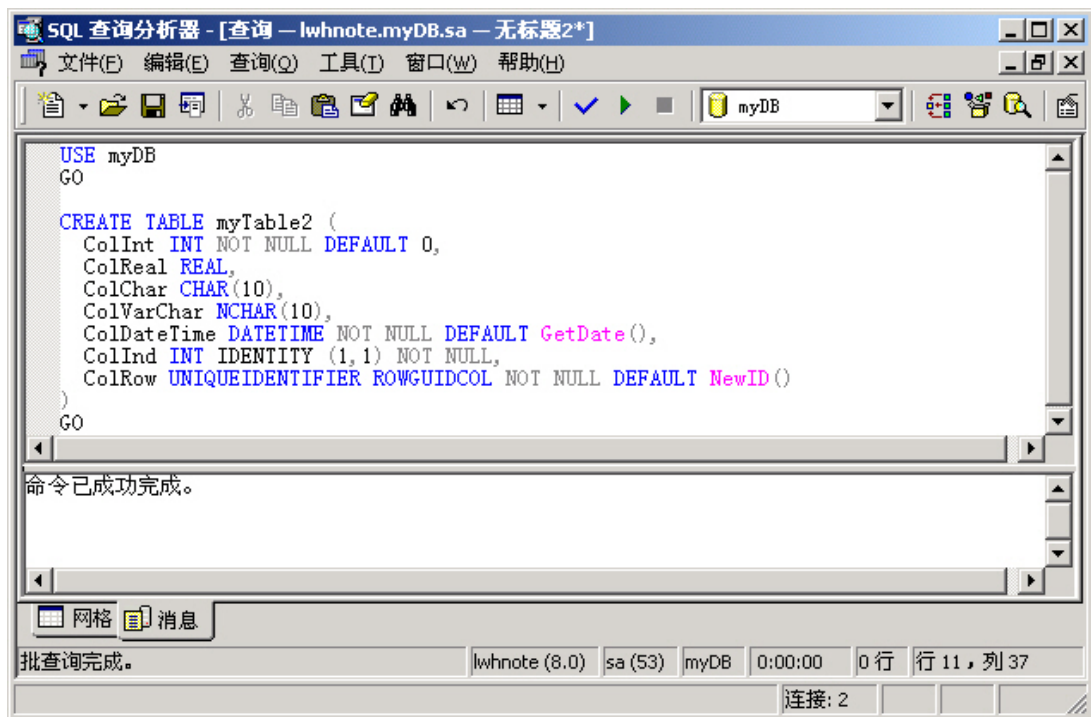
自动生成了数据表 myTable 的 SQL 语句。



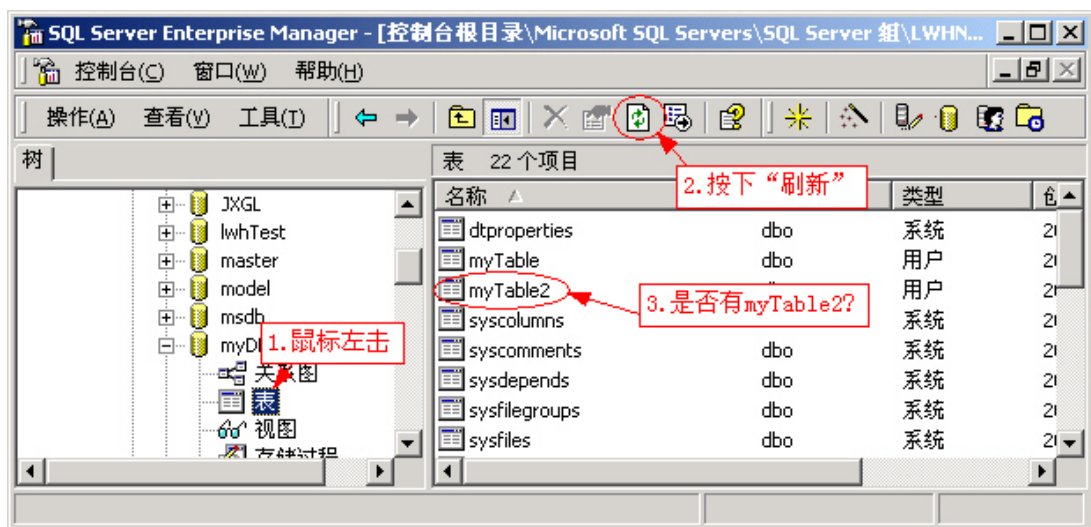
二、用查询分析器创建数据表

 帮助索引：<创建表>

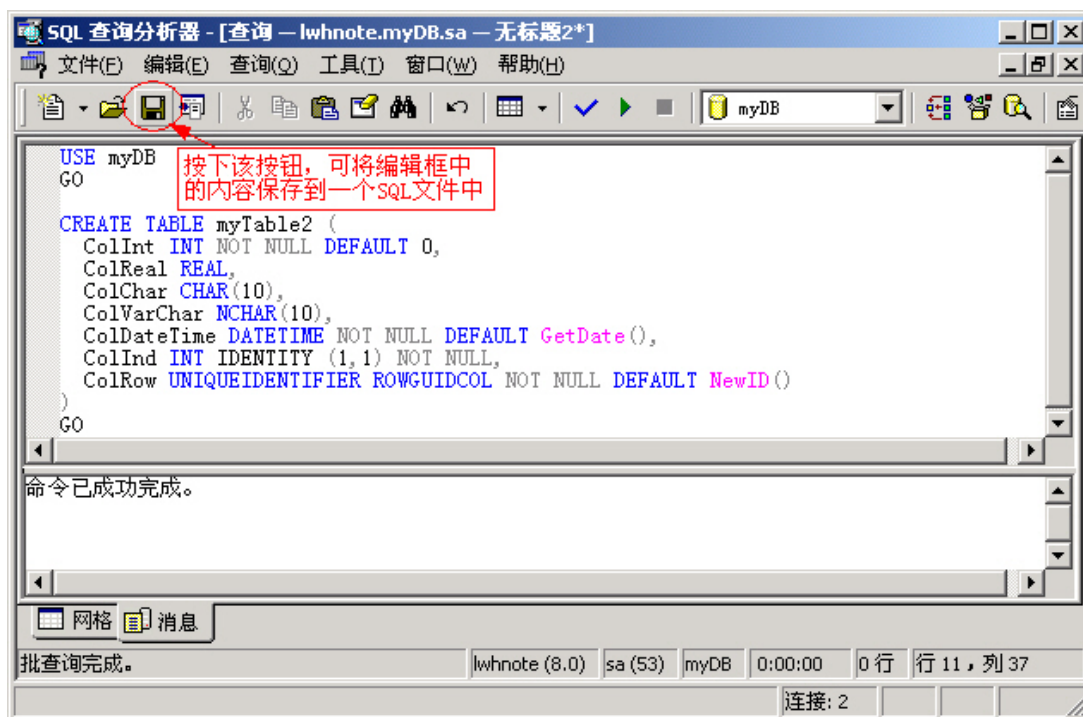
例 1：创建上面用企业管理器创建的数据表，但表名为 myTable2。



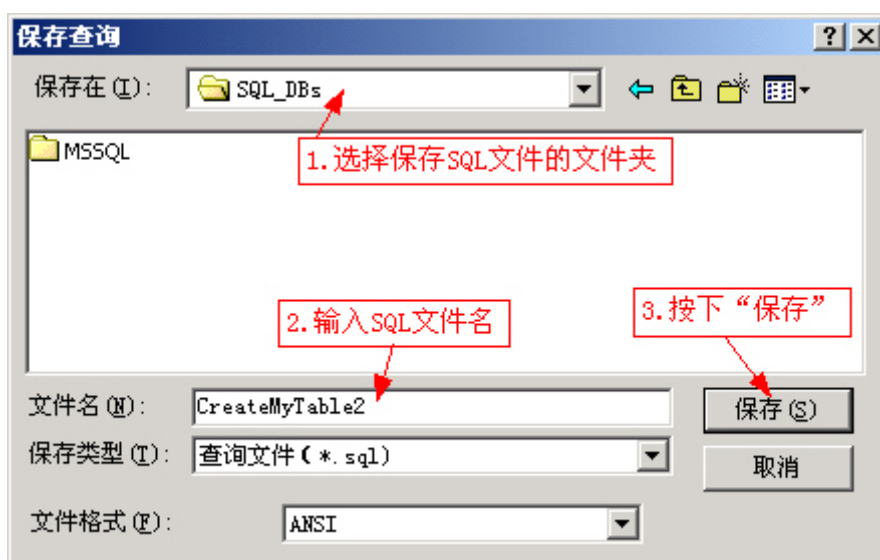
到企业管理器中检验数据表 myTable2 是否已经建立。



另外，可以将查询分析器中的内容保存到一个 SQL 文件，便于以后使用。

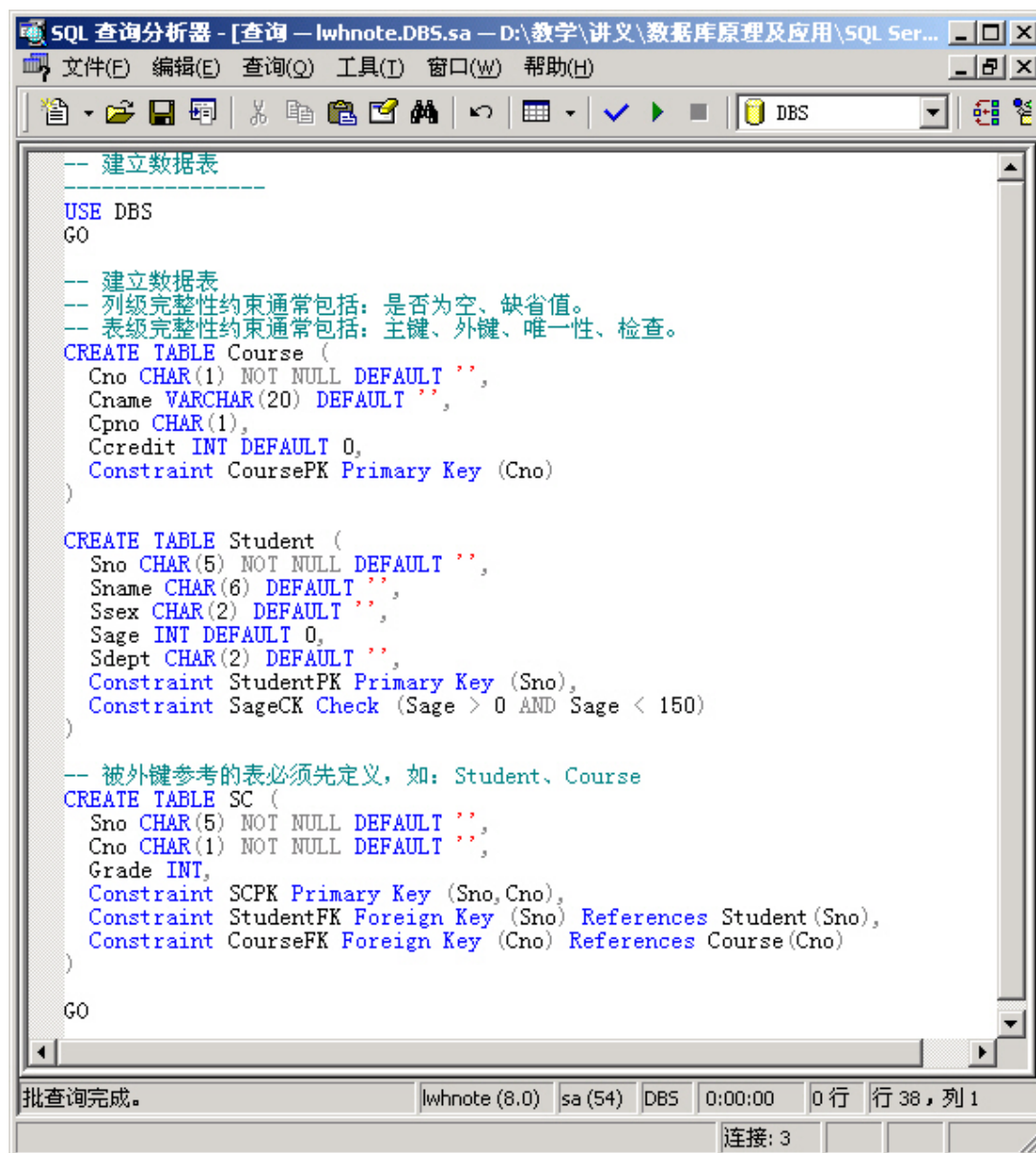


按下保存按钮后，弹出指定 SQL 文件名的对话框



按下“保存”，即可保存当前内容到一个指定的 SQL 文件中。

例 2：创建学生成绩管理系统数据表，包括三个数据表：课程数据表 Course、学生数据表 Student、学生成绩数据表 SC。



```
-- 建立数据表
USE DBS
GO

-- 建立数据表
-- 列级完整性约束通常包括：是否为空、缺省值。
-- 表级完整性约束通常包括：主键、外键、唯一性、检查。
CREATE TABLE Course (
    Cno CHAR(1) NOT NULL DEFAULT '',
    Cname VARCHAR(20) DEFAULT '',
    Cpno CHAR(1),
    Ccredit INT DEFAULT 0,
    Constraint CoursePK Primary Key (Cno)
)

CREATE TABLE Student (
    Sno CHAR(5) NOT NULL DEFAULT '',
    Sname CHAR(6) DEFAULT '',
    Ssex CHAR(2) DEFAULT '',
    Sage INT DEFAULT 0,
    Sdept CHAR(2) DEFAULT '',
    Constraint StudentPK Primary Key (Sno),
    Constraint SageCK Check (Sage > 0 AND Sage < 150)
)

-- 被外键参考的表必须先定义，如：Student、Course
CREATE TABLE SC (
    Sno CHAR(5) NOT NULL DEFAULT '',
    Cno CHAR(1) NOT NULL DEFAULT '',
    Grade INT,
    Constraint SCPK Primary Key (Sno, Cno),
    Constraint StudentFK Foreign Key (Sno) References Student (Sno),
    Constraint CourseFK Foreign Key (Cno) References Course (Cno)
)

GO
```

批查询完成。 | lwhtnote (8.0) | sa (54) | DBS | 0:00:00 | 0 行 | 行 38, 列 1 | 连接: 3

注：上面这个建立学生成绩管理系统的命令集可以保存到一个 SQL 文件中（如：CreateTable.sql），便于以后使用。

2.1.3 创建带约束的数据库表

约束(Constraint)是 Microsoft SQL Server 提供的自动保持数据库完整性的一种方法，定义了可输入表或表的单个列中的数据限制条件。在 SQL Server 中有 5 种约束：主关键字约束(Primary Key Constraint)、外关键字约束(Foreign Key Constraint)、唯一性约束(Unique Constraint)、检查约束(Check Constraint) 和 缺省约束(Default Constraint)。

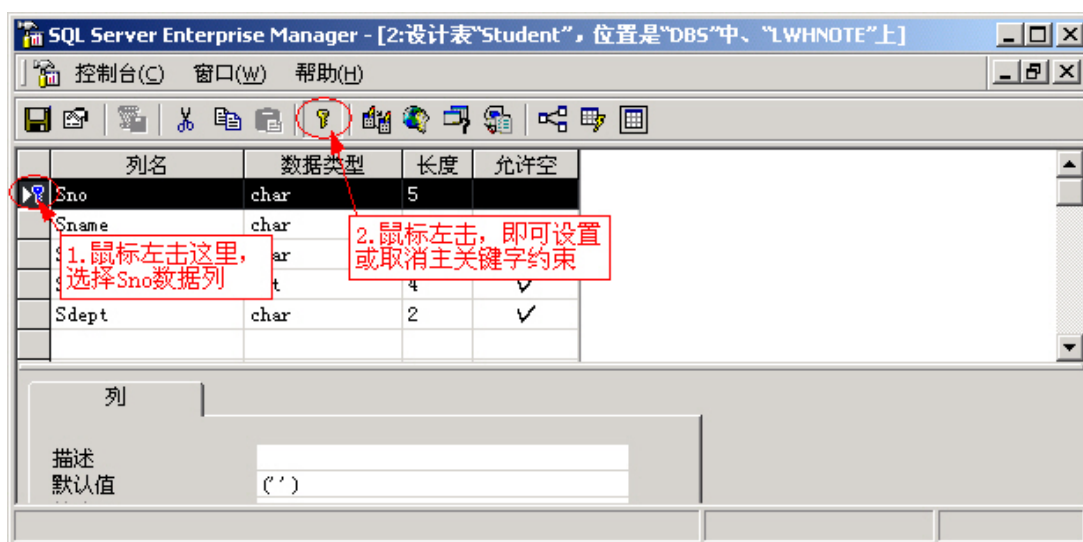
一、主关键字约束

帮助索引：<约束：PRIMARY KEY>

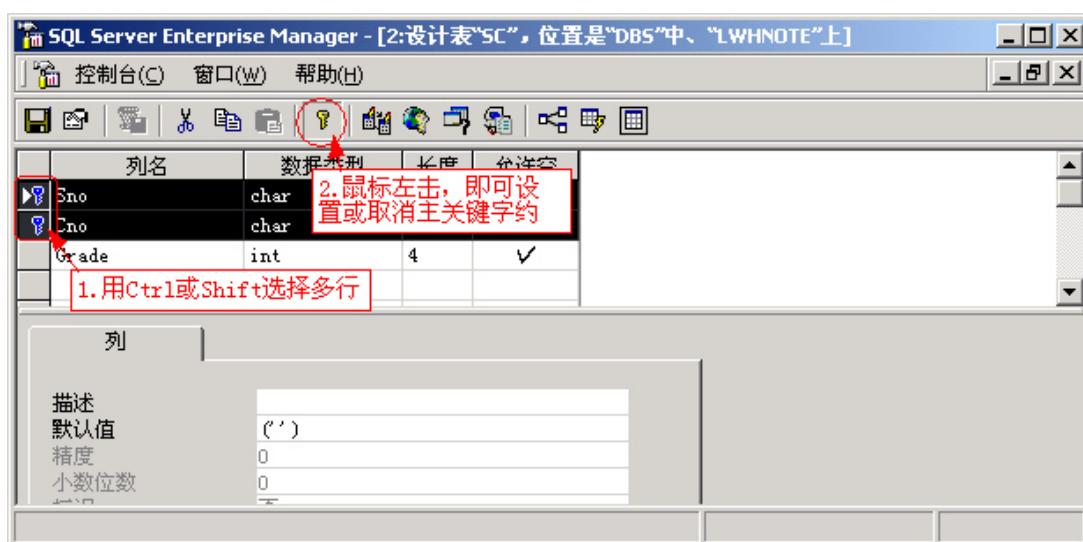
主关键字约束指定表的一列或几列的组合的值在表中具有惟一性，即能惟一地指定一行记录。每个表中只能有一个主关键字约束，且 IMAGE 和 TEXT 类型的列不能被指定为主关键字，也不允许指定主关键字列有 NULL 属性。

A. 用企业管理器指定主关键字约束

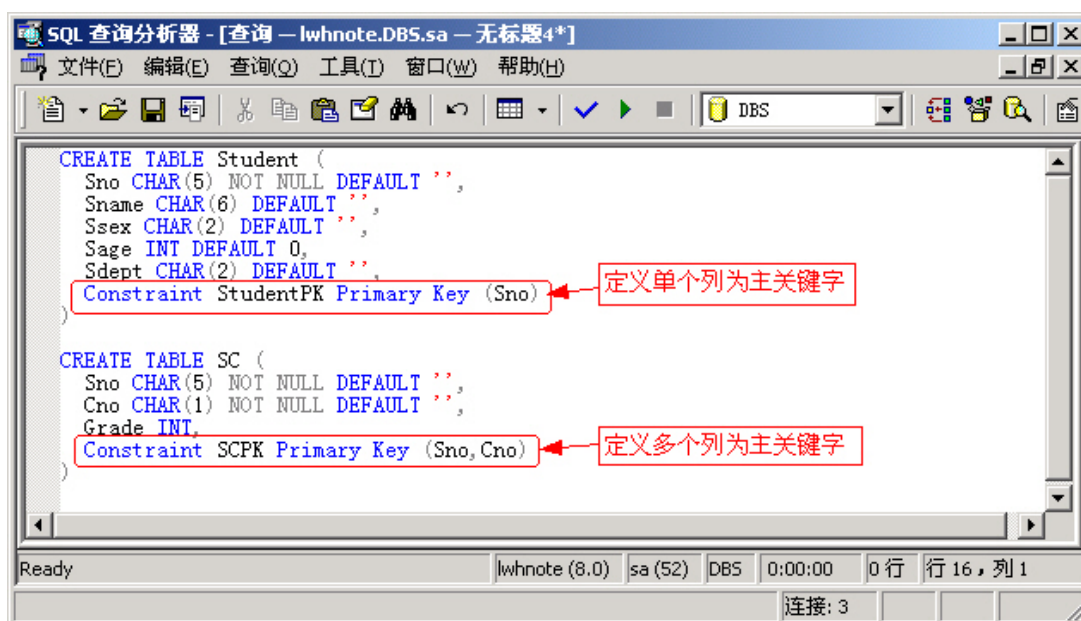
例 1：设置 Student 数据表的 Sno 为主关键字约束



例 2：设置 SC 数据表的 Sno 和 Cno 为主关键字约束



B. 用查询分析器指定主关键字约束



二、外关键字约束

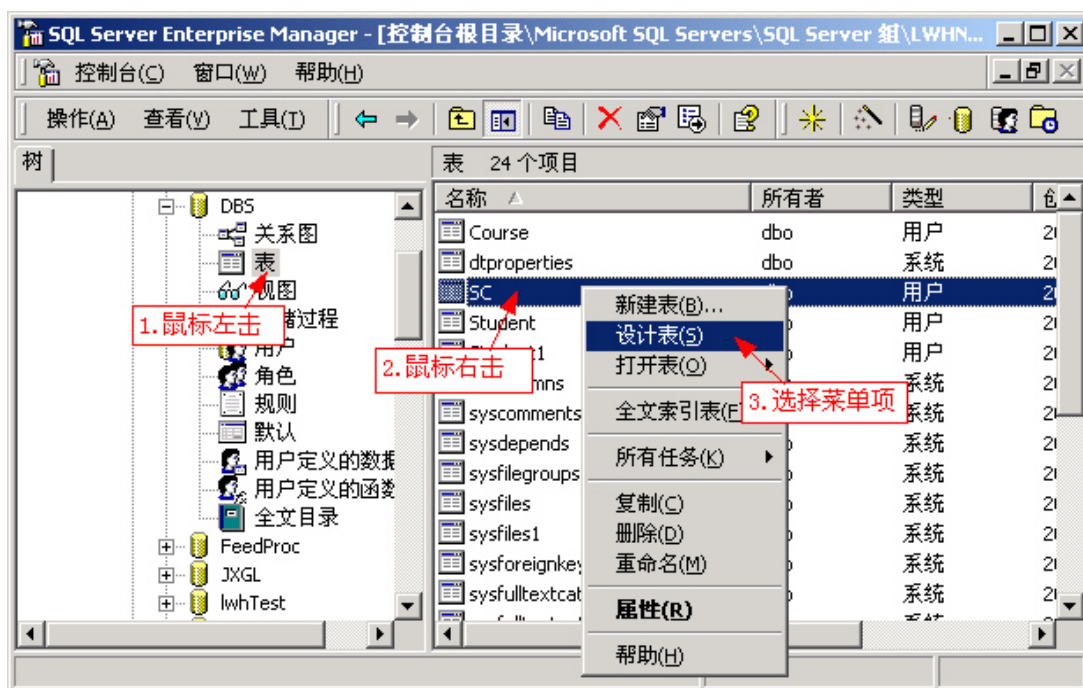
帮助索引: <约束: FOREIGN KEY>

外关键字约束定义了表之间的关系, 当一个表中的一个列或多个列的组合和其它表中的主关键字定义相同时, 就可以将这些列或列的组合定义为外关键字, 并设定它适合哪个表中哪些列相关联。这样, 当在定义主关键字约束的表中更新列值时, 其它表中有与之相关联的外关键字约束的表中的外关键字列也将被相应地做相同的更新。外关键字约束的作用还体现在, 当向含有外关键字的表插入数据时, 如果与之相关联的表的列中无与插入的外关键字列值相同的值时, 系统会拒绝插入数据。与主关键字相同, 不能使用一个定义为 TEXT 或 IMAGE 数据类型的列创建外关键字。外关键字最多由 16 个列组成。

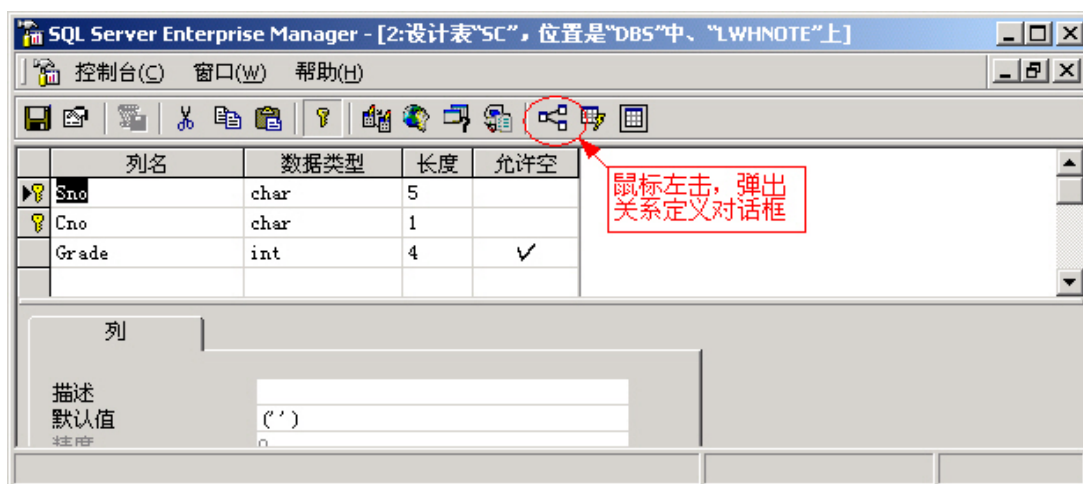
A. 用企业管理器指定外关键字约束

如: 设置 SC 数据表的 Sno 是一个外关键字, 参考 Student 表中的 Sno; 设置 SC 数据表的 Cno 是一个外关键字, 参考 Course 表中的 Cno。

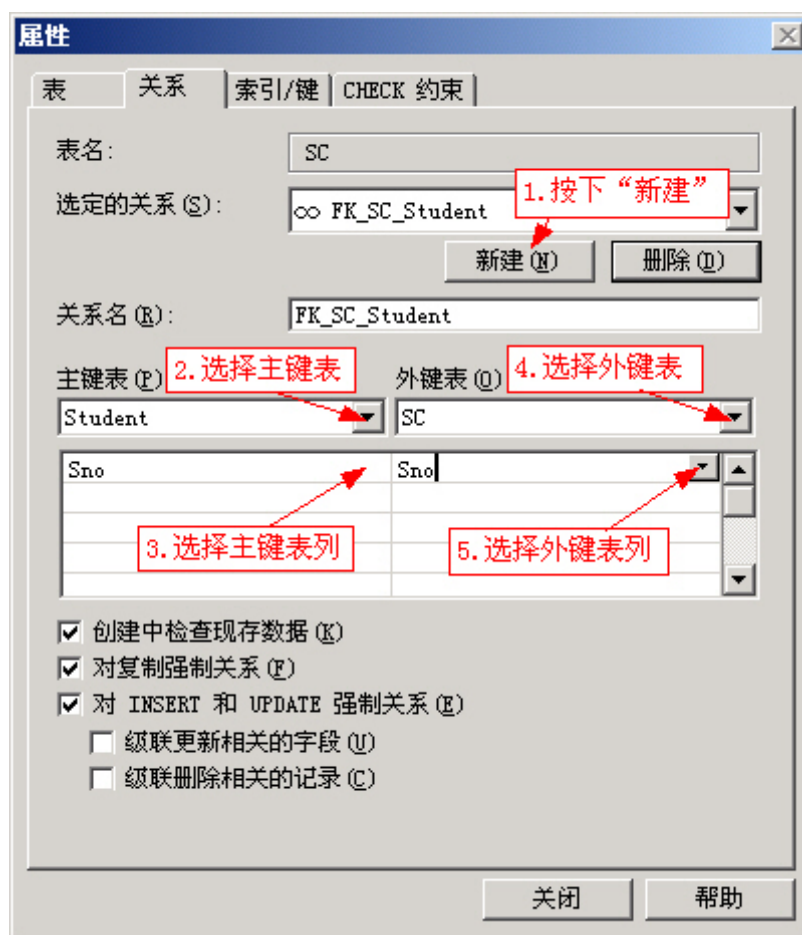
下图操作, 打开数据表 SC 的数据结构



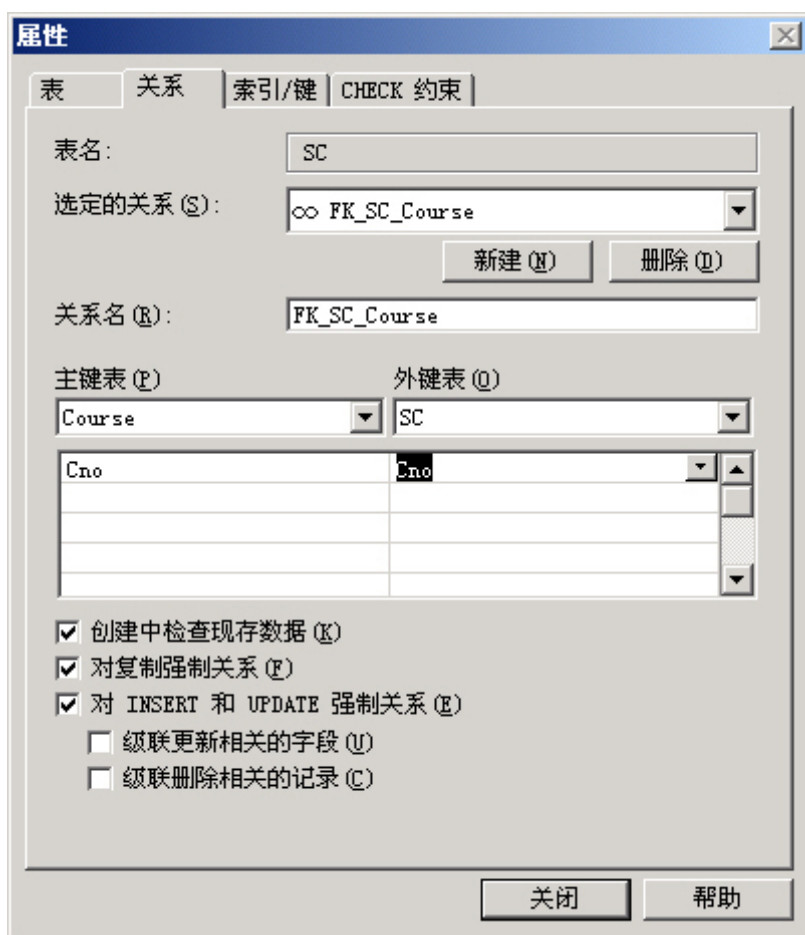
下图为数据表 SC 的数据结构



下图为数据表 SC 的关系，演示了设置 SC 数据表的 Sno 是一个外关键字，参考 Student 表中的 Sno

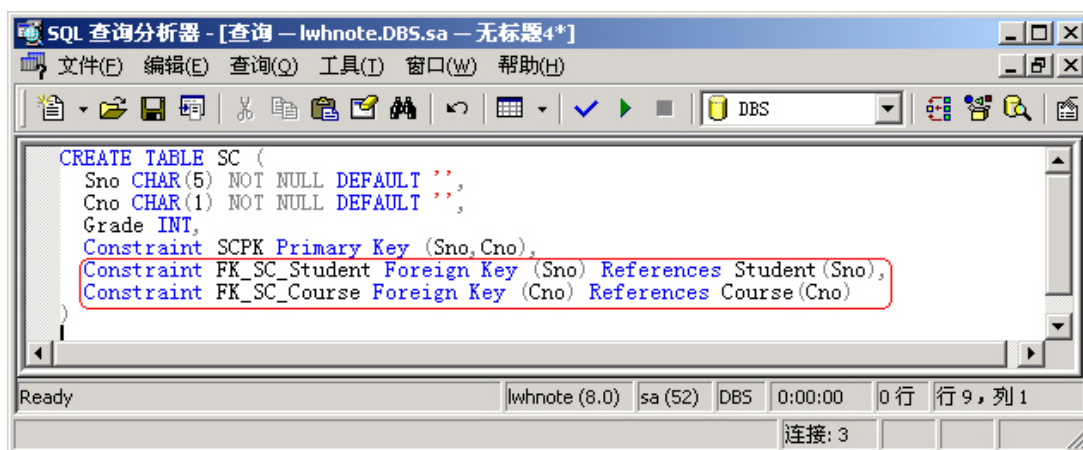


下图设置 SC 数据表的 Cno 是一个外关键字，参考 Course 表中的 Cno



B. 用查询分析器指定外关键字约束

如：设置 SC 数据表的 Sno 是一个外关键字，参考 Student 表中的 Sno；设置 SC 数据表的 Cno 是一个外关键字，参考 Course 表中的 Cno。



三、唯一性约束

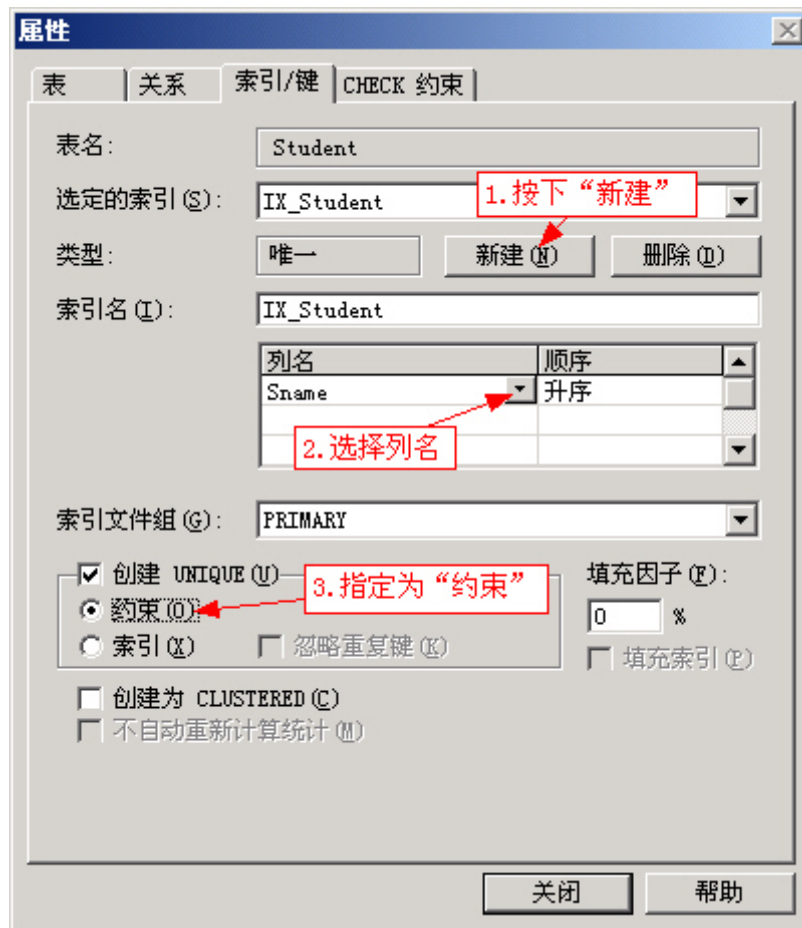
 帮助索引：<约束： UNIQUE>

可使用 **UNIQUE** 约束确保在非主键列或允许为空值的列中不输入重复值。尽管

UNIQUE 约束和 PRIMARY KEY 约束都强制唯一性，但对于非主键列或允许为空值的列应使用 UNIQUE 约束而不是 PRIMARY KEY 约束。

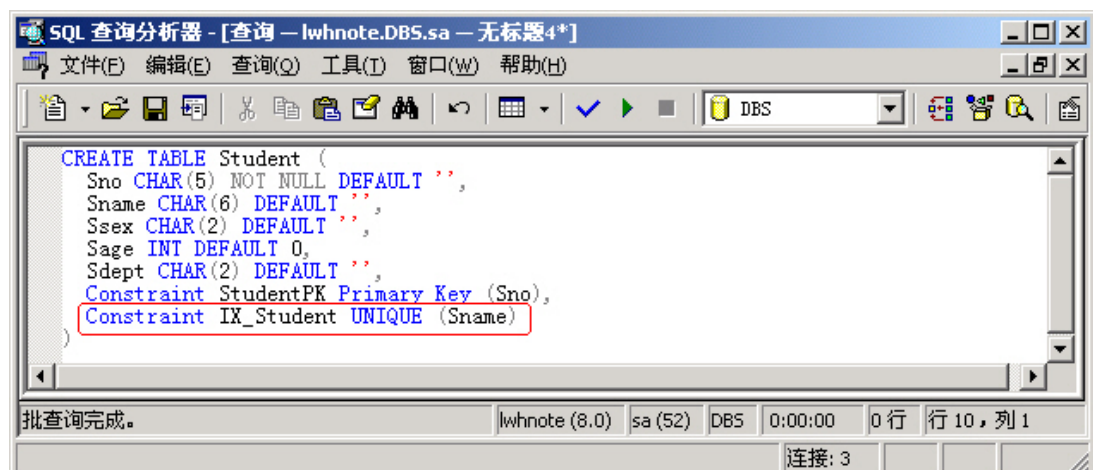
A. 用企业管理器指定唯一性约束

如：设置 Student 数据表的 Sname 是一个满足唯一性约束的列。



B. 用查询分析器指定唯一性约束

如：设置 Student 数据表的 Sname 是一个满足唯一性约束的列。



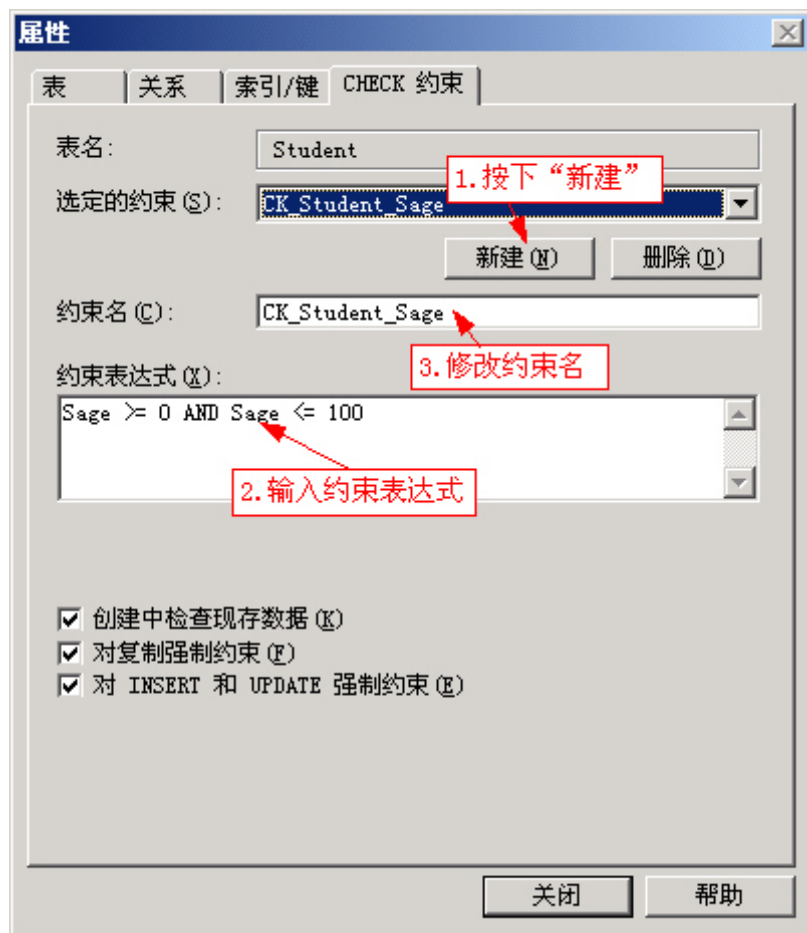
四、检查约束

 帮助索引: <约束: CHECK>

检查约束对输入列或整个表中的值设置检查条件,以限制输入值,保证数据库的数据完整性。可以对每个列设置检查条件。

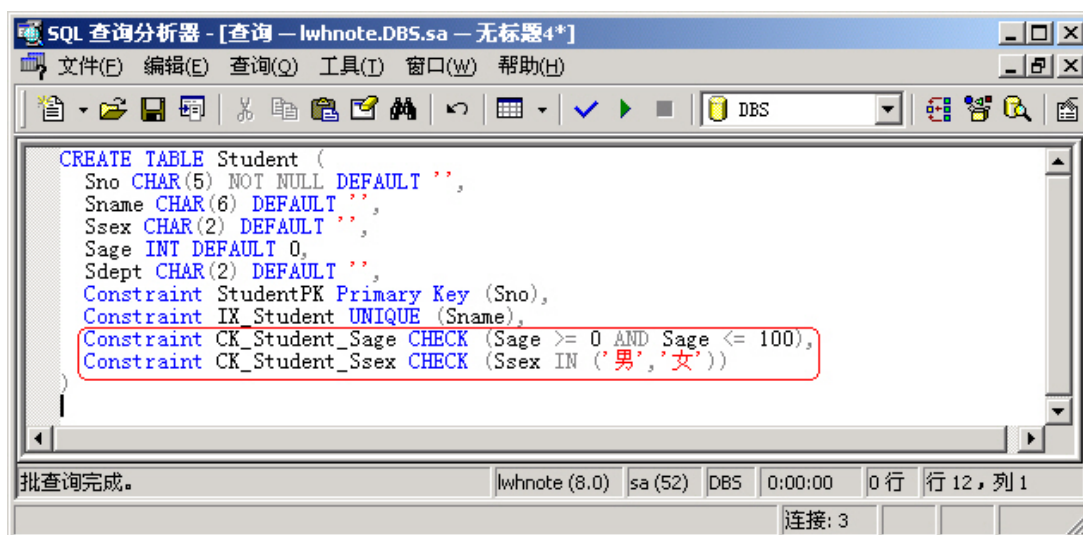
A. 用企业管理器指定唯一性约束

如: 设置 Student 数据表中的 Sage 的值必须在 0-100 之间。



B. 用查询分析器指定唯一性约束

如: 设置 Student 数据表中的 Sage 的值必须在 0-100 之间, Ssex 的值只能是“男”或“女”。



五、缺省约束

帮助索引: <默认约束>

缺省约束通过定义列的缺省值或使用数据库的缺省值对象绑定表的列，来指定列的缺省值。SQL Server 推荐使用缺省约束，而不使用定义缺省值的方式来指定列的缺省值。

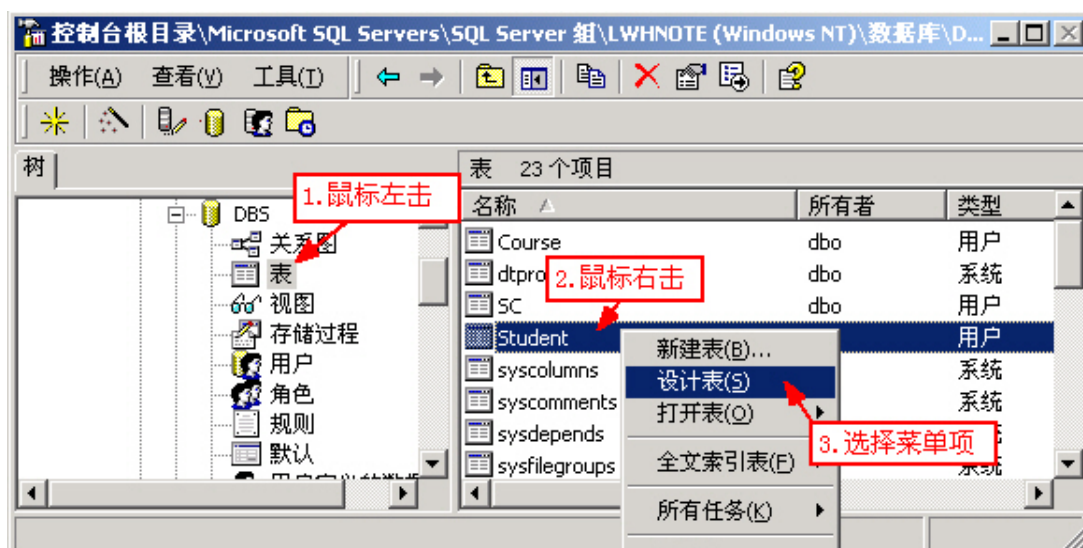
2.1.4 修改数据库表

当数据库表创建好后，可以根据需要对表的列、约束等属性进行添加、删除及修改，这就需要修改数据表的结构。

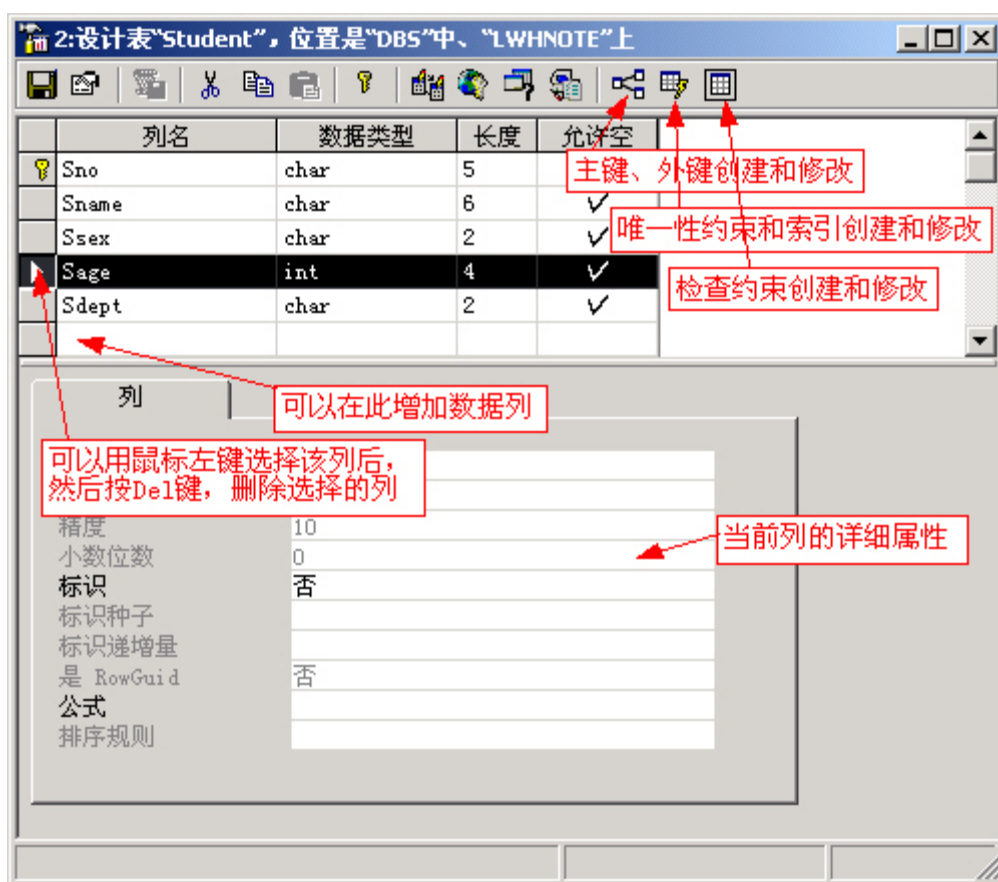
一、使用企业管理器修改数据库表

帮助索引: <修改表>

若要修改某数据表，必须首先进行数据表的结构设计界面，然后就与新建数据表一样，可以增加、删除数据列，修改某数据列的数据类型、长度等属性，还可以修改数据表的主键、外键、索引、唯一性约束、检查约束等。如：修改学生数据表 Student。



下图显示了 Student 数据表的结构

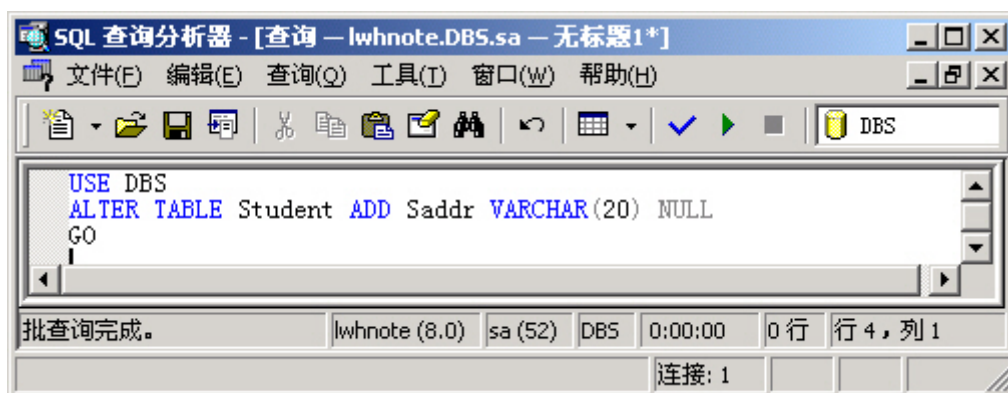


二、使用查询分析器修改数据库表

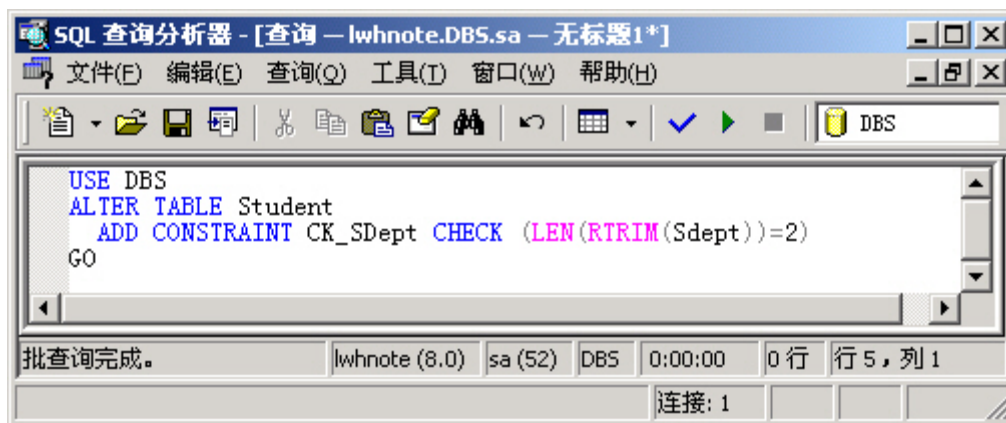
帮助索引: <修改表>

使用查询分析器修改数据库表在实际工作中往往意义不大，实际工作中往往不如首先删除要修改的数据表，然后重新创建数据表。

例 1: 向学生数据表 Student 中增加一个数据列住址 Saddr，字符型 20。

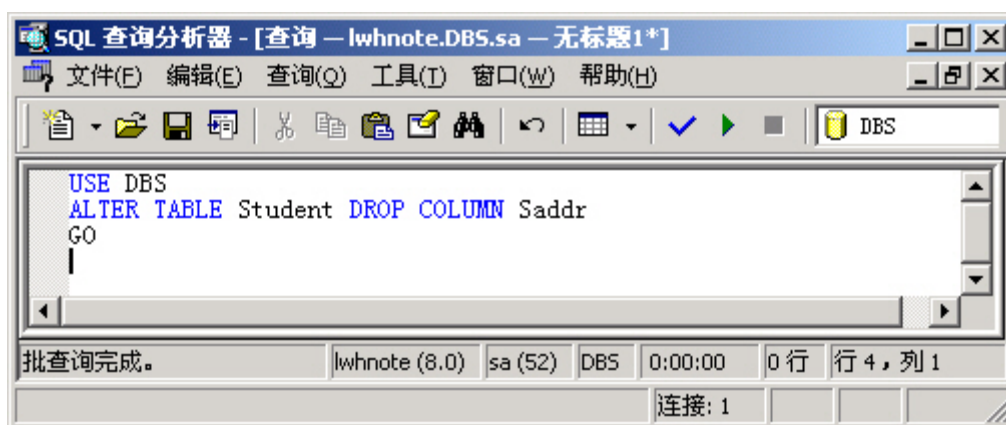


例 2: 给学生数据表 Student 中系号增加一个检查约束，检查其字符数必须为 2。



注：在计算字符串长度时，一个 ASCII 字符占一个长度，而一个汉字也只占一个长度，这点与其它系统有点不一样，如 VC、VB、Delphi 中，一个汉字都是占二个长度。

例 3：删除学生数据表 Student 中住址列。



2.1.5 更新数据库表的数据

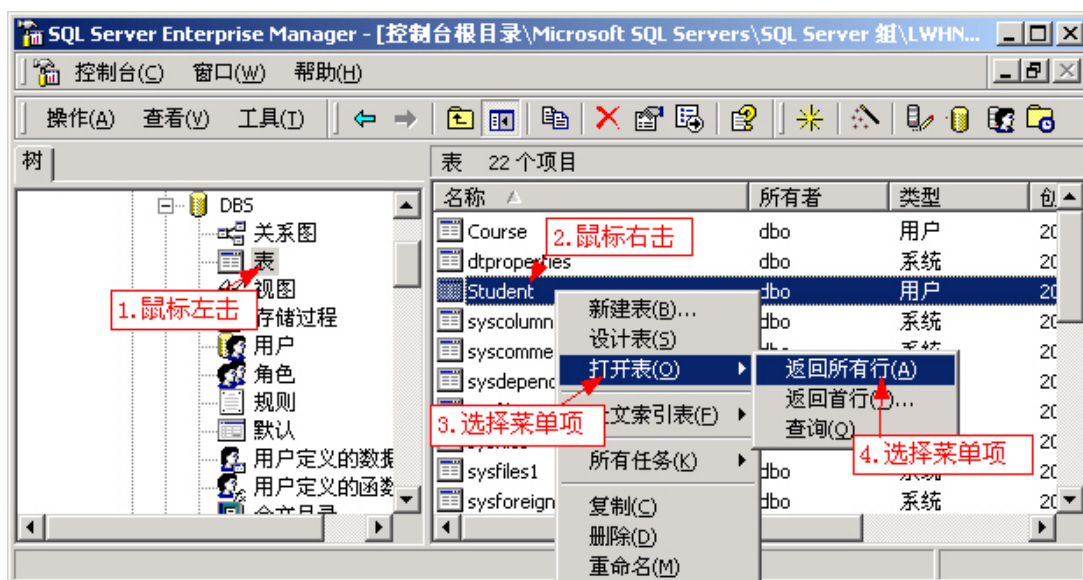
数据库表的更新包括：插入、修改和删除三种操作。

一、插入数据

 帮助索引：<添加行>

A. 用企业管理器插入数据

例 1：向学生数据表 Student 插入数据

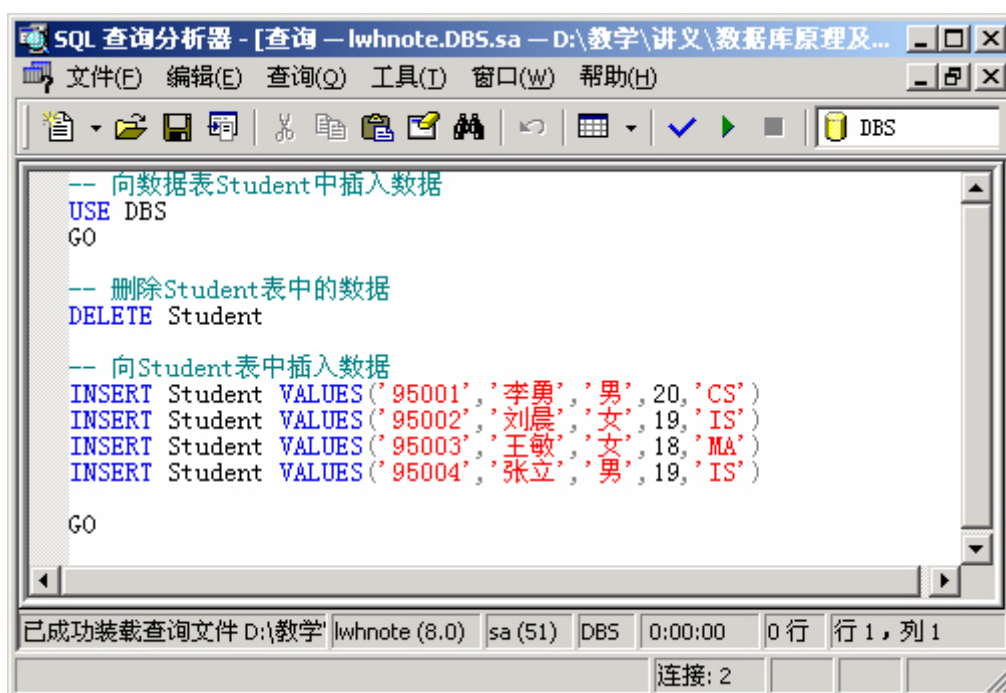


下图显示了插入数据后的情况

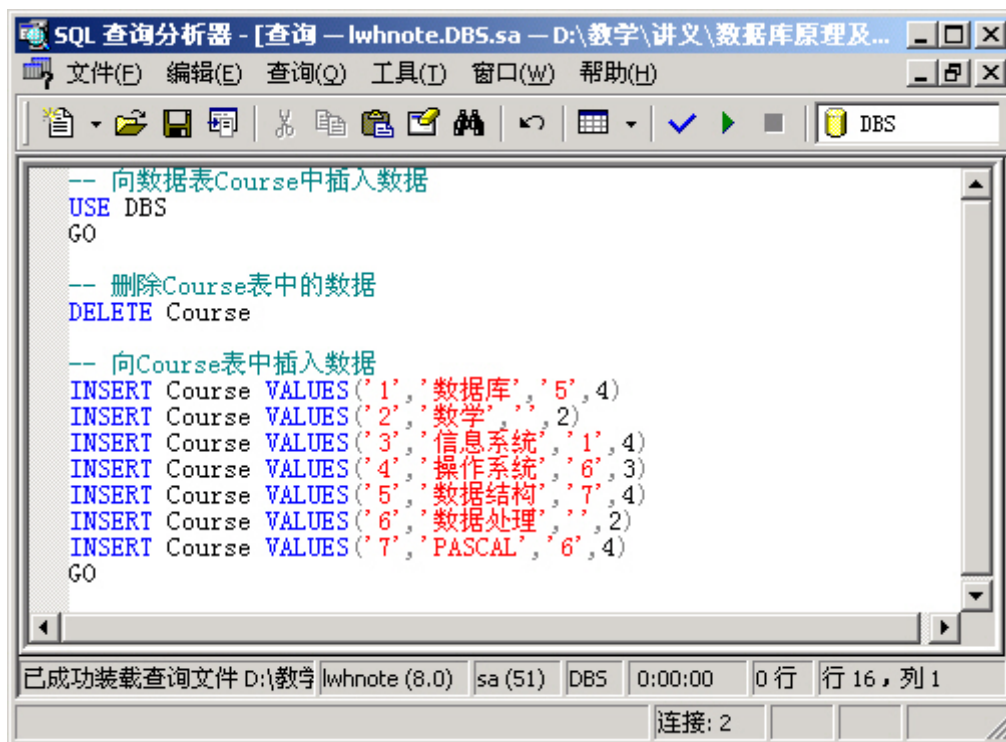
Sno	Sname	Ssex	Sage	Sdept
95001	李勇	男	20	CS
95002	刘晨	女	19	IS
95003	王敏	女	18	MA
95004	张立	男	19	IS

B. 用查询分析器插入数据

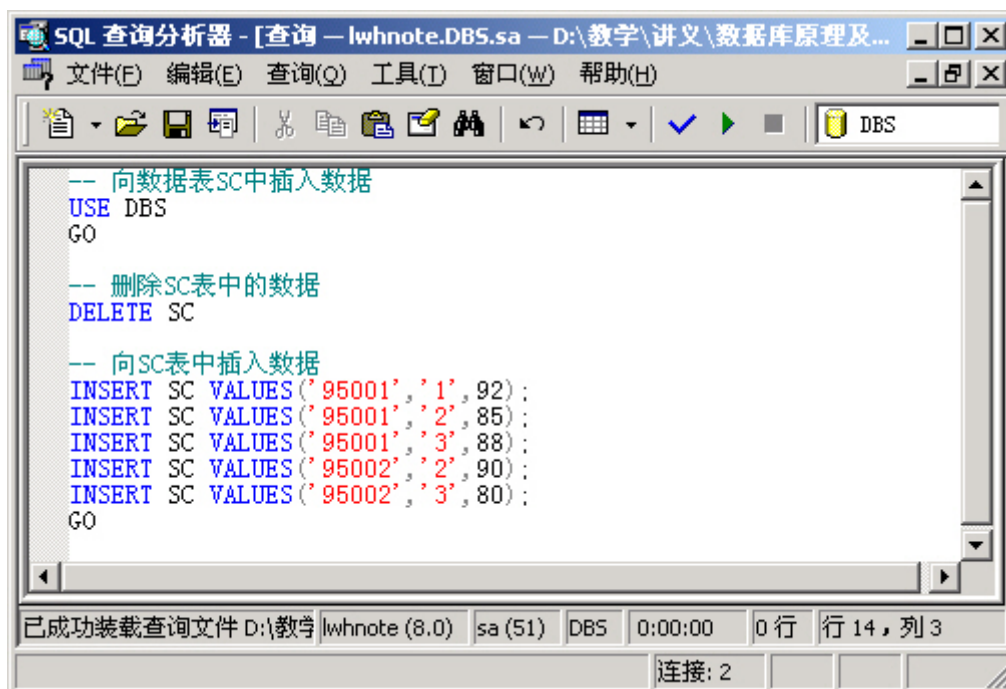
例 1: 向学生数据表 Student 插入数据, 其数据同上。



例 2：向课程数据表 Course 插入数据。

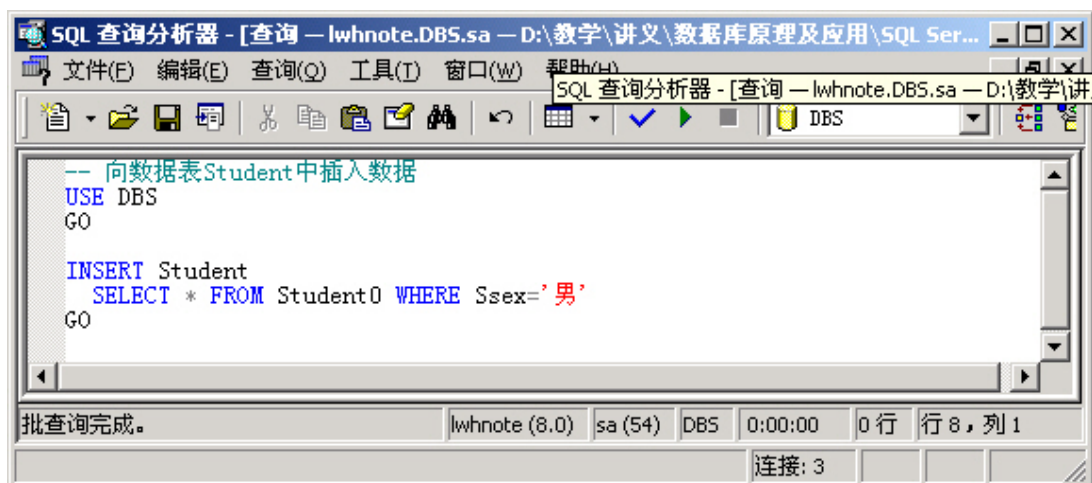


例 3：向学生成绩数据表 SC 插入数据。



注：上面几个向学生成绩管理系统中增加数据的命令集可以保存到不同的 SQL 文件中（如：InsStudent.sql、InsCourse.sql、InsSC.sql），便于以后使用。

例 4：将另一个学生数据表 Student0 中男生的数据插入到学生数据表 Student 中。



二、修改数据

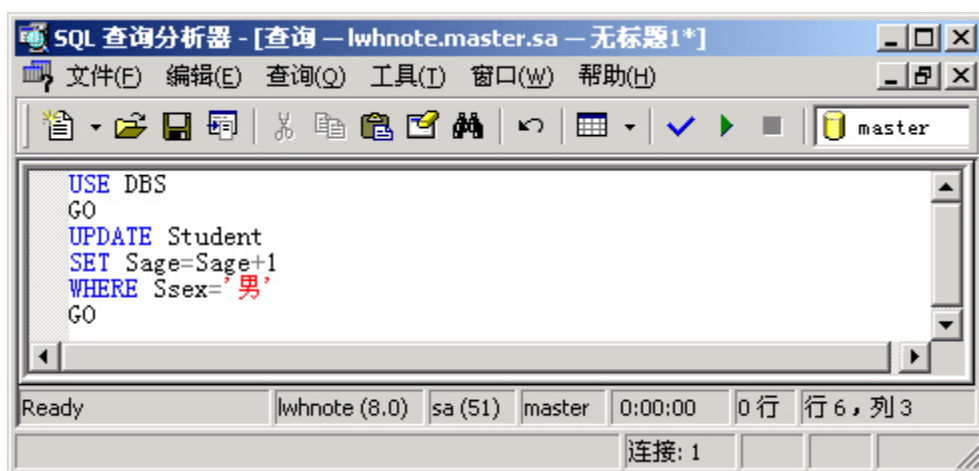
 帮助索引: <修改数据|Update>

A. 用企业管理器修改数据

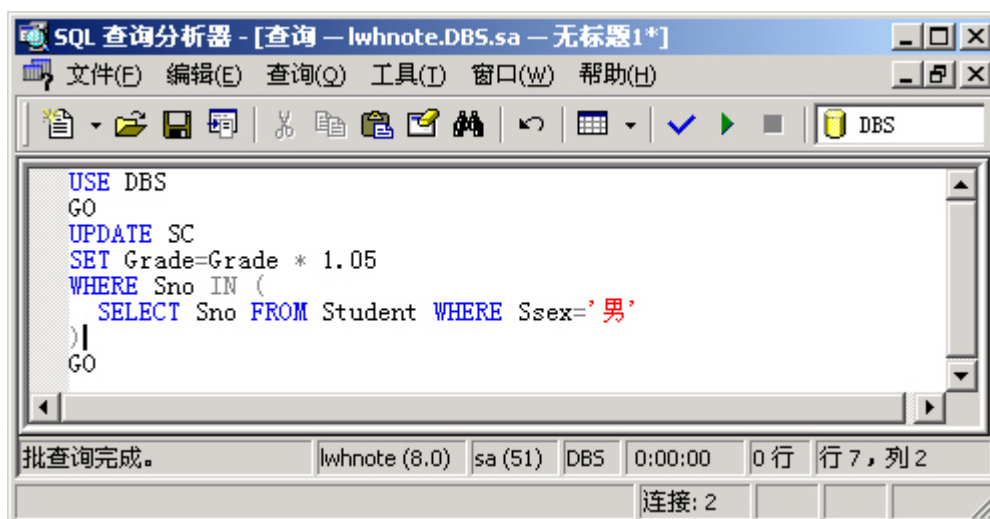
参见<[用企业管理器插入数据](#)>, 在数据库表窗口中, 可以修改数据表数据。

B. 用查询分析器修改数据

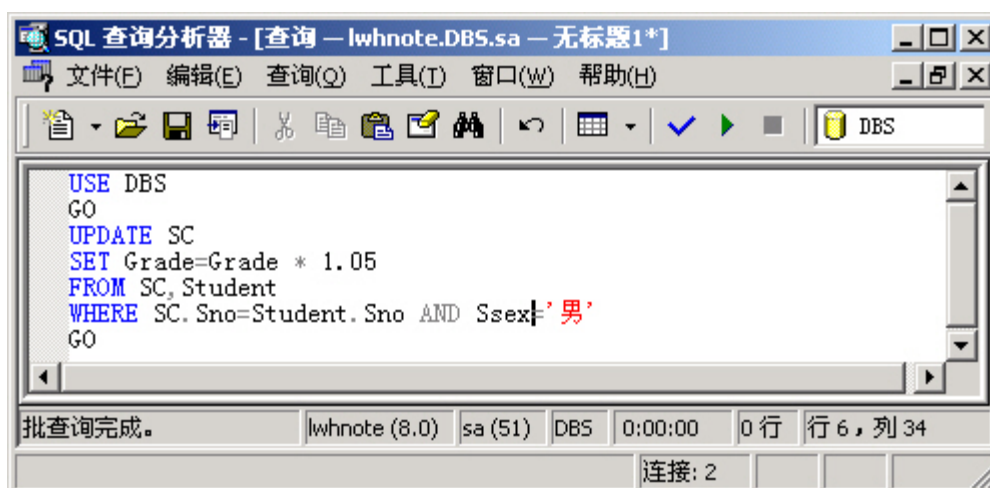
例 1: 将学生数据表 Student 男同学的年龄加 1。



例 2: 将成绩数据 SC 男同学的成绩增加 5% (使用子查询)。



例 3：将成绩数据 SC 男同学的成绩增加 5%（使用数据表连接）。

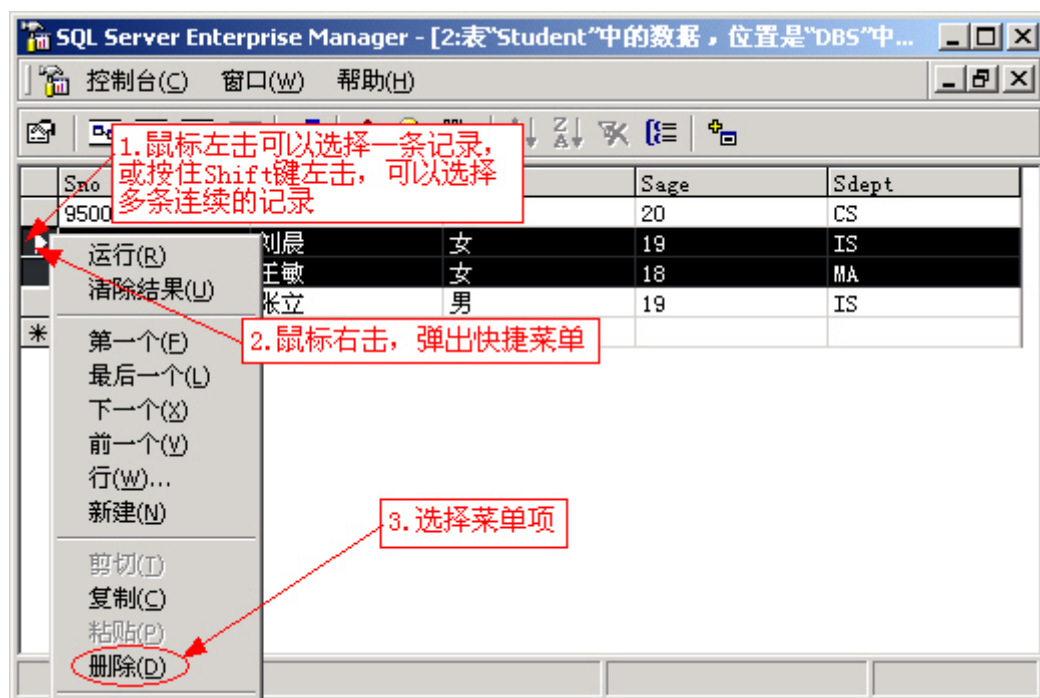


三、删除数据

 帮助索引：<删除数据|使用 Delete 语句删除行>

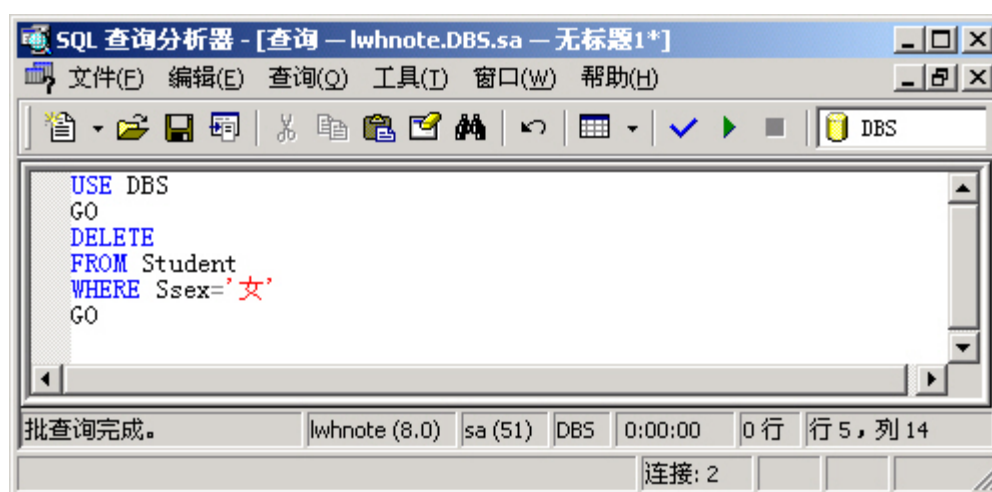
A. 用企业管理器删除数据

例 1：删除学生数据表 Student 中女同学数据。

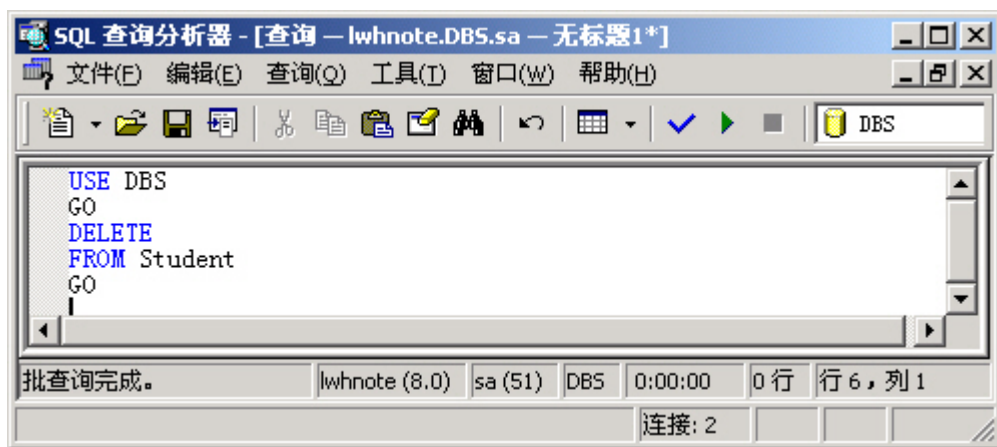


B. 用查询分析器删除数据

例 1：删除学生数据表 Student 中女同学数据。



例 2：删除学生数据表 Student 的全部数据。



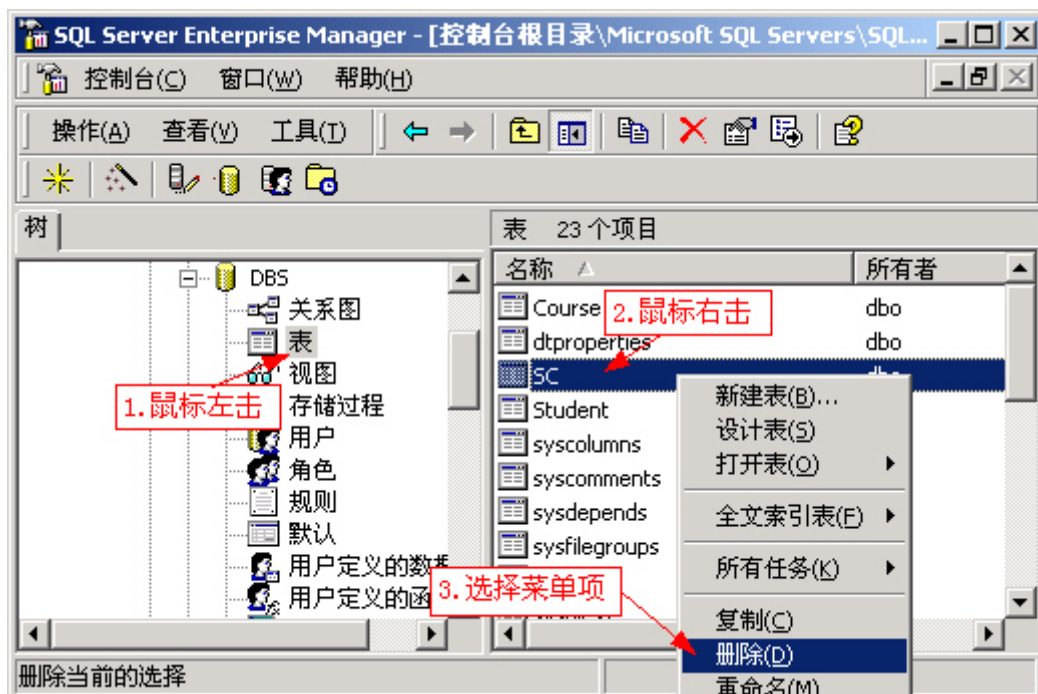
2.1.6 删除数据库表

帮助索引: <删除表|删除表>

注: 在删除数据表时, 必须首先删除与此表有外关键字约束的表。如: 学生数据表 Student 的 Sno 是学生成绩表 SC 的外关键字, 删除 Student 之前, 必须首先删除 SC。

一、用企业管理器删除数据表

如: 删除学生成绩数据表 SC

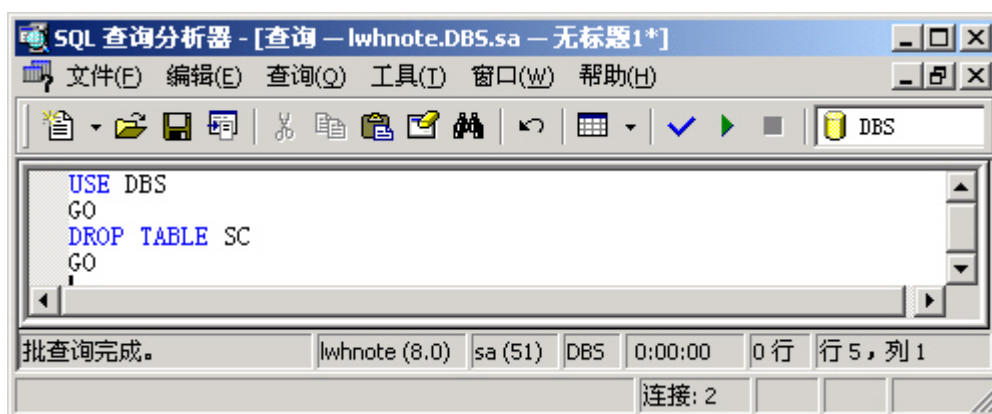


弹出删除对象选择框, 按下“全部除去”即可删除指定的数据表



二、用查询分析器删除数据表

如：删除学生成绩数据表 SC



2.2 索引

在使用数据库的过程中，接触最多的就是数据库中的表。表是数据存储的地方，是数据库中最重要的一部分。管理好表也就管理好了数据库。

用户对数据库最频繁的操作是进行数据查询，一般情况下，数据库在进行查询操作时需要对整个表进行数据搜索。当表中的数据很多时，搜索数据就需要很长的时间，这就造成了服务器的资源浪费。为了提高检索数据的能力，数据库引入了索引机制。本章将介绍索引的概念及其创建与管理。

2.2.1 创建索引

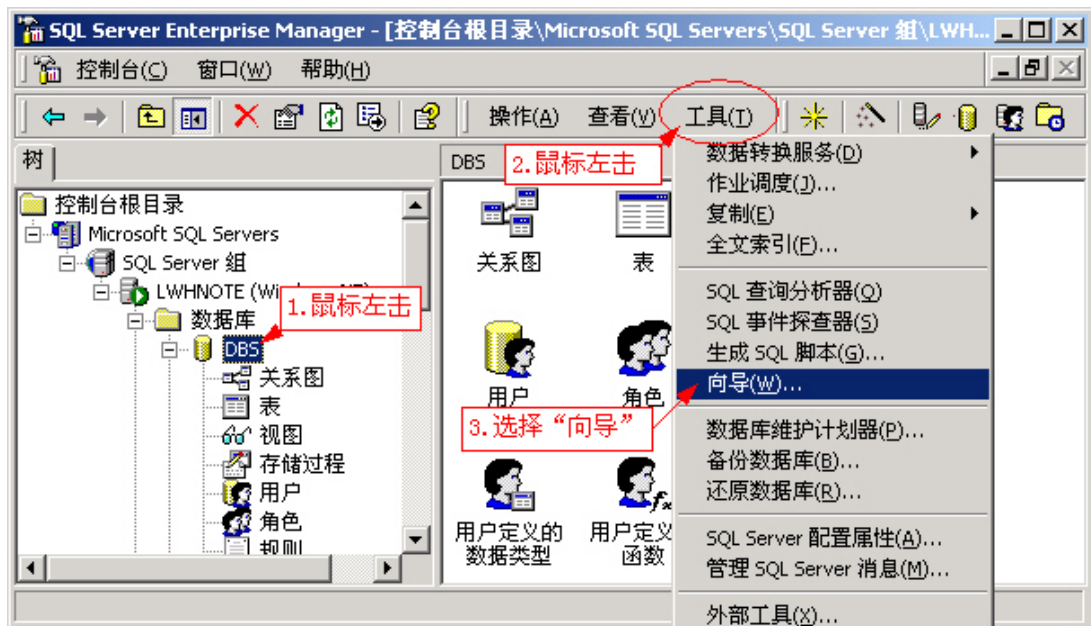
 帮助索引：<创建索引|概述|创建索引>

一、用企业管理器创建索引

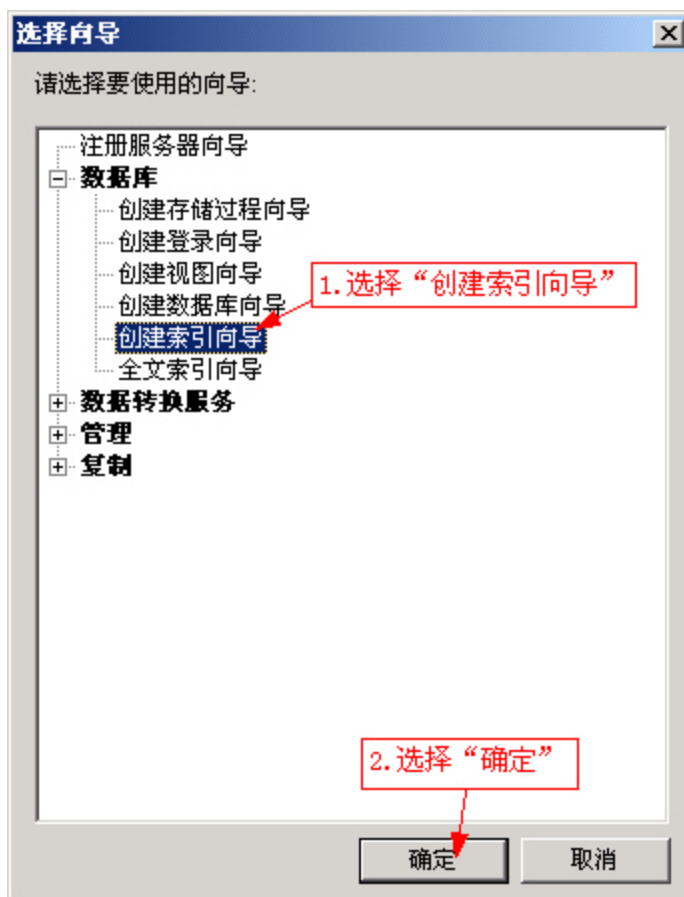
A. 用创建索引向导创建索引

 帮助索引：<创建索引向导>

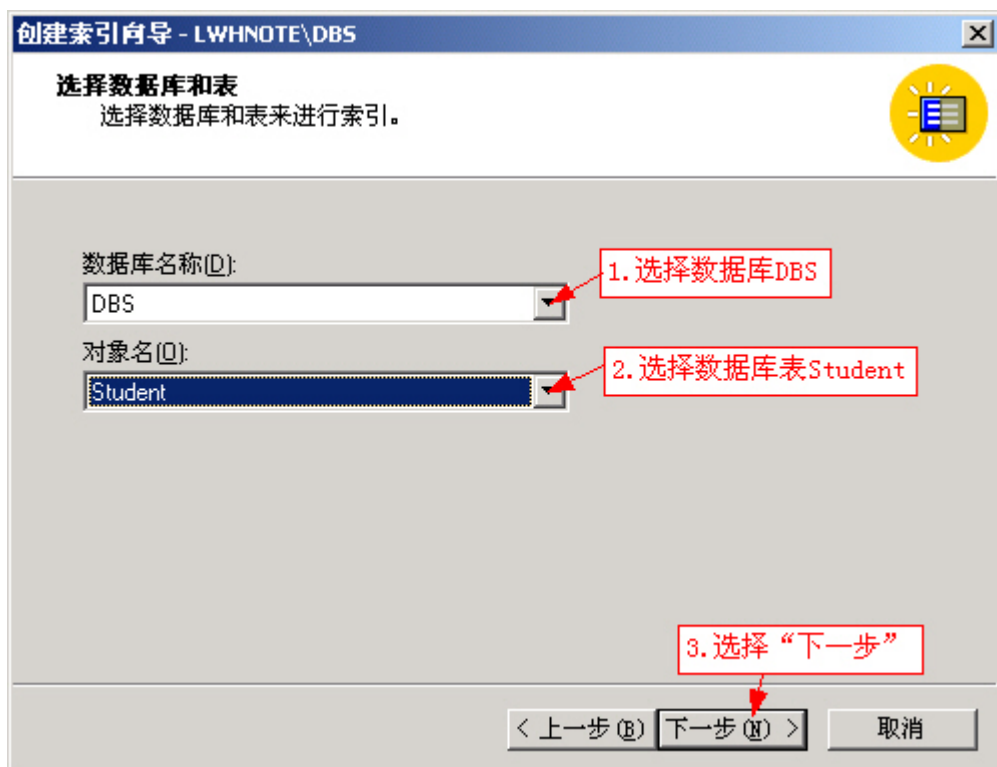
例：对学生数据表 Student 中的姓名 Sname 建立一个索引。



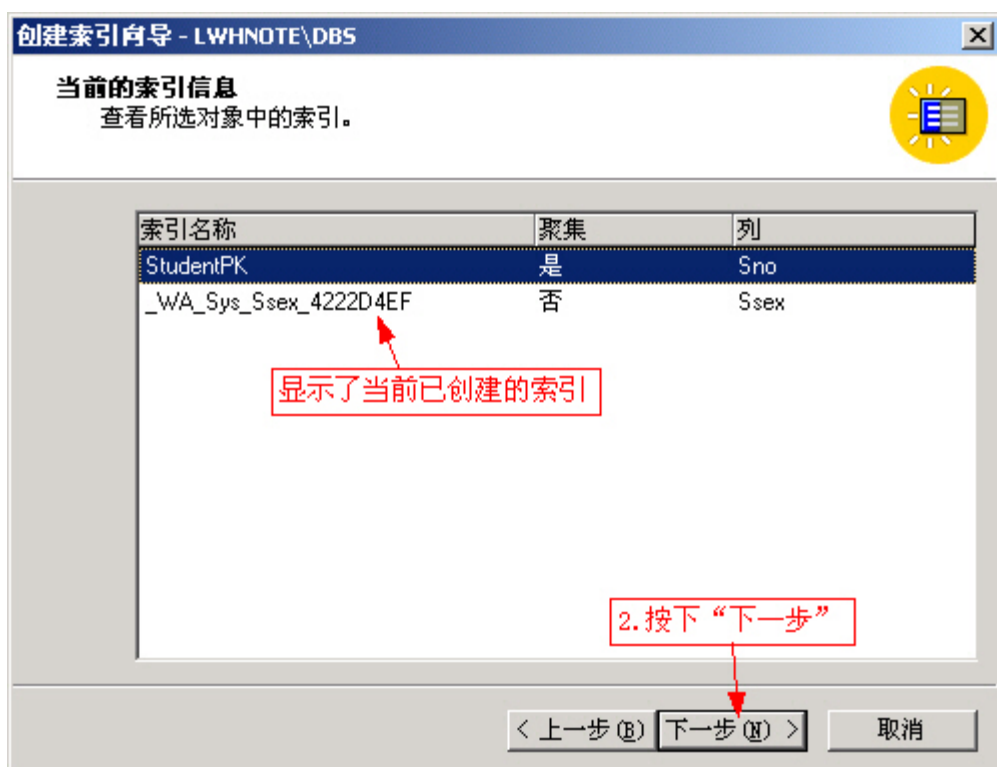
弹出如下对话框



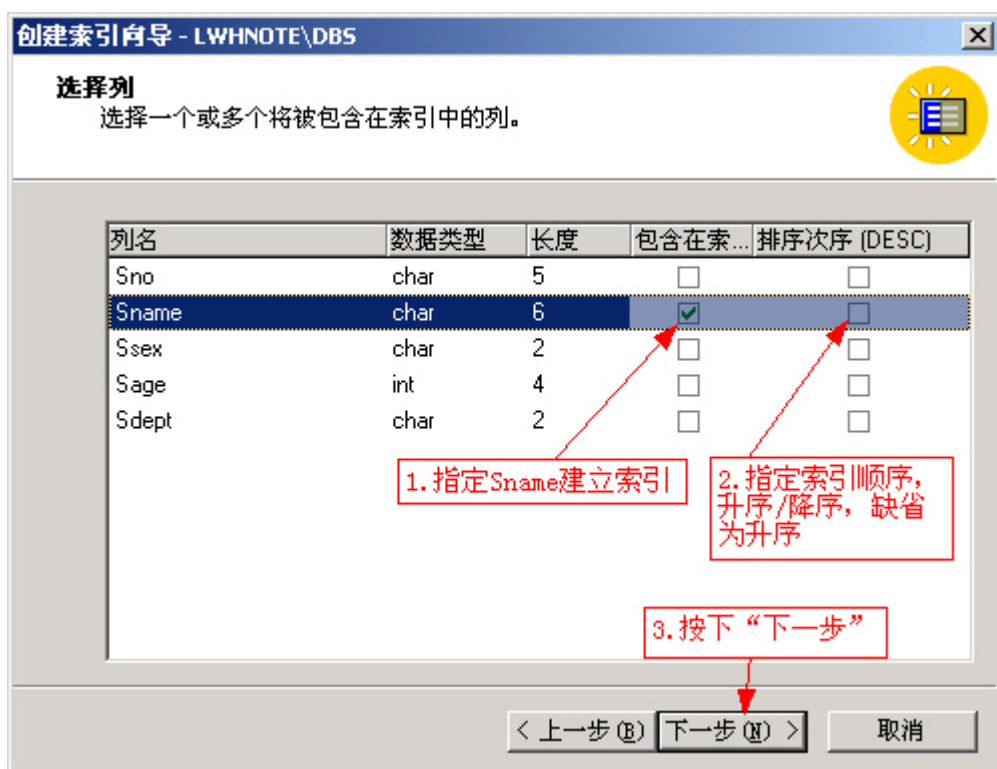
按“下一步”，直到出现如下对话框



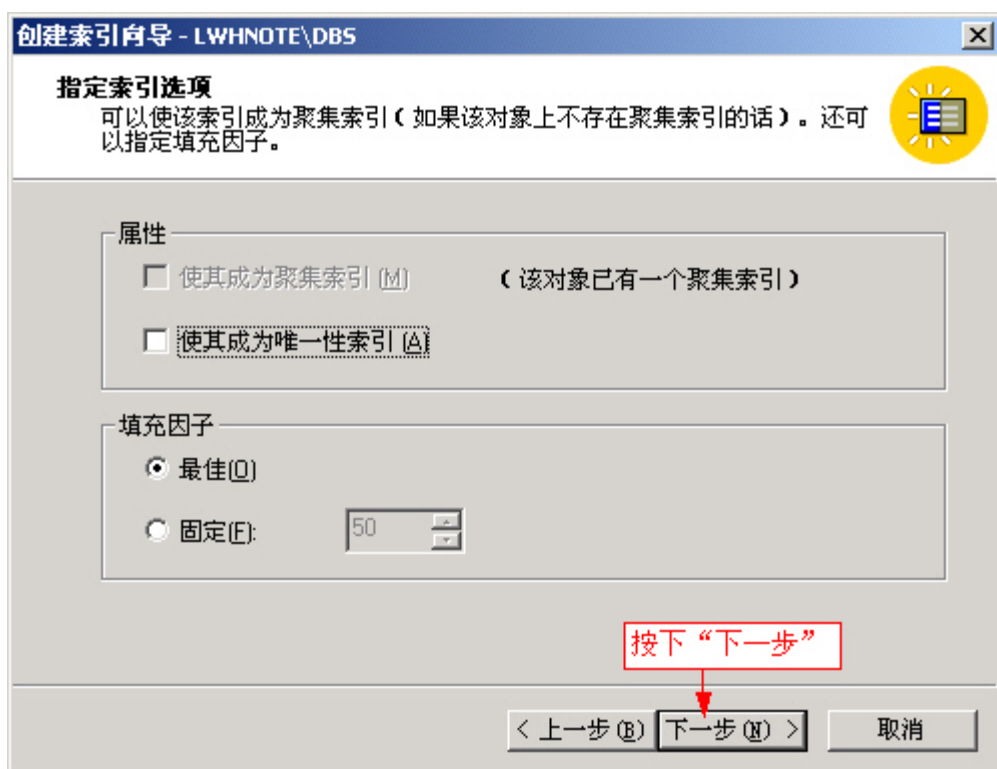
在下面的对话中，显示了当前已经建立的索引



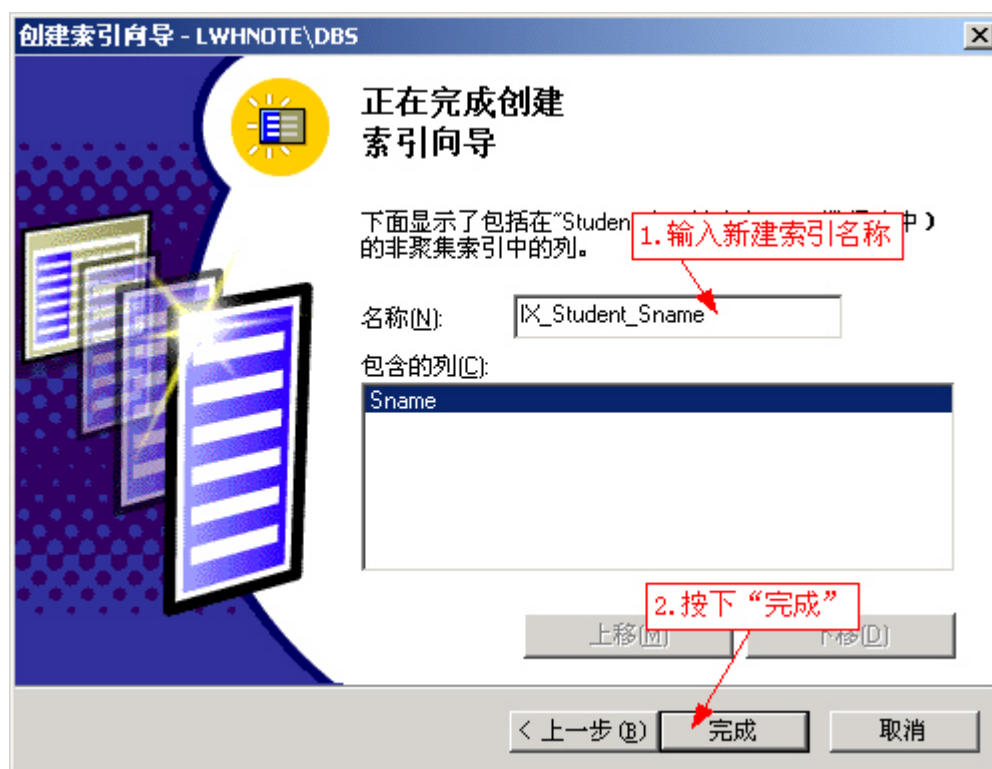
在下面对话框中，指定新建索引的列名



指定新建索引的选项

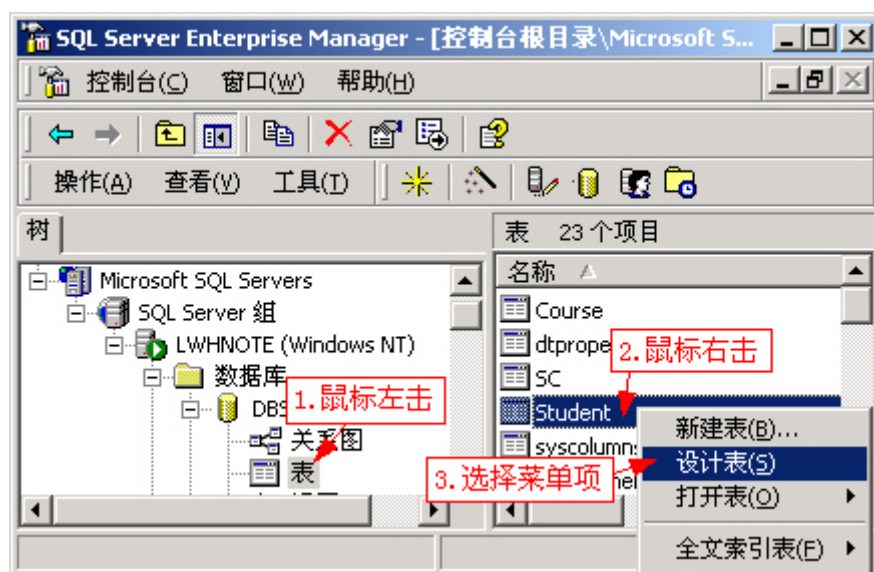


按“完成”，创建索引结束。

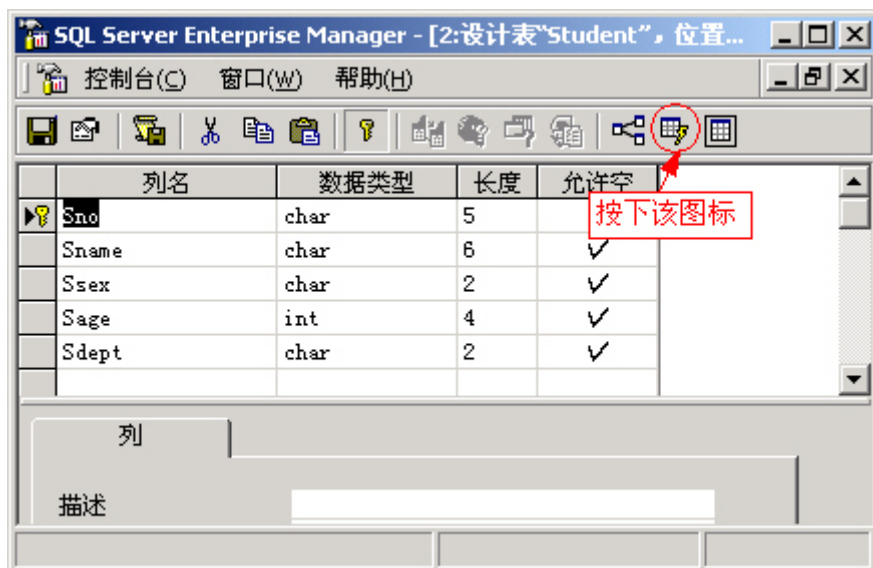


B. 用表的属性对话框创建索引

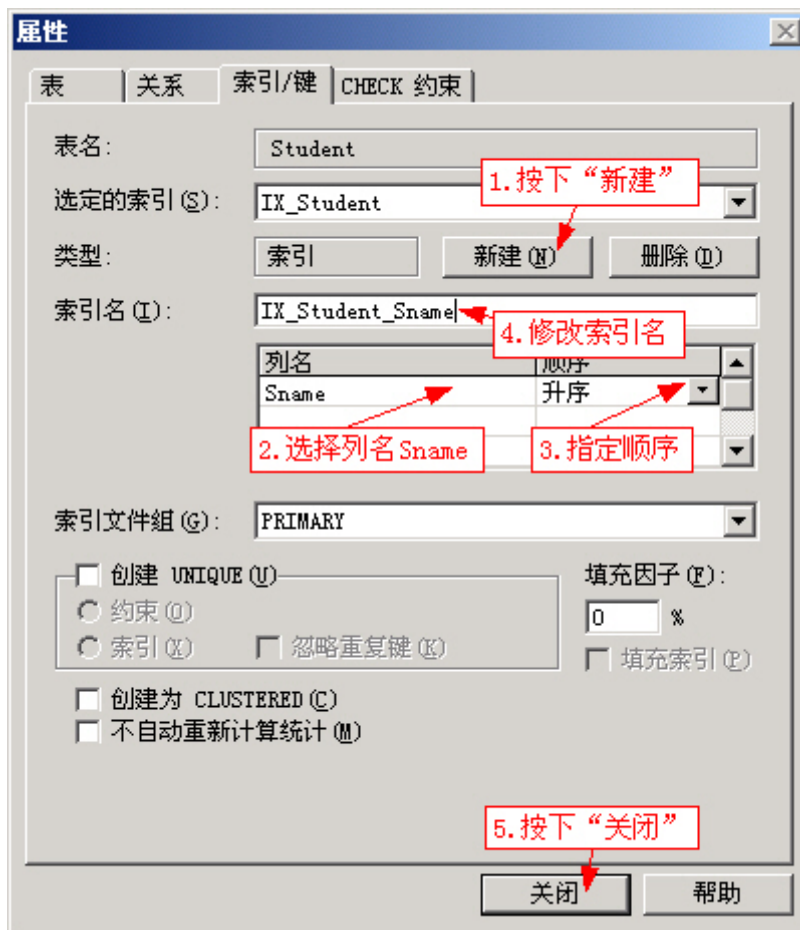
例：对学生数据表 Student 中的姓名 Sname 建立一个索引。



在下面对话框中，按下索引属性按钮

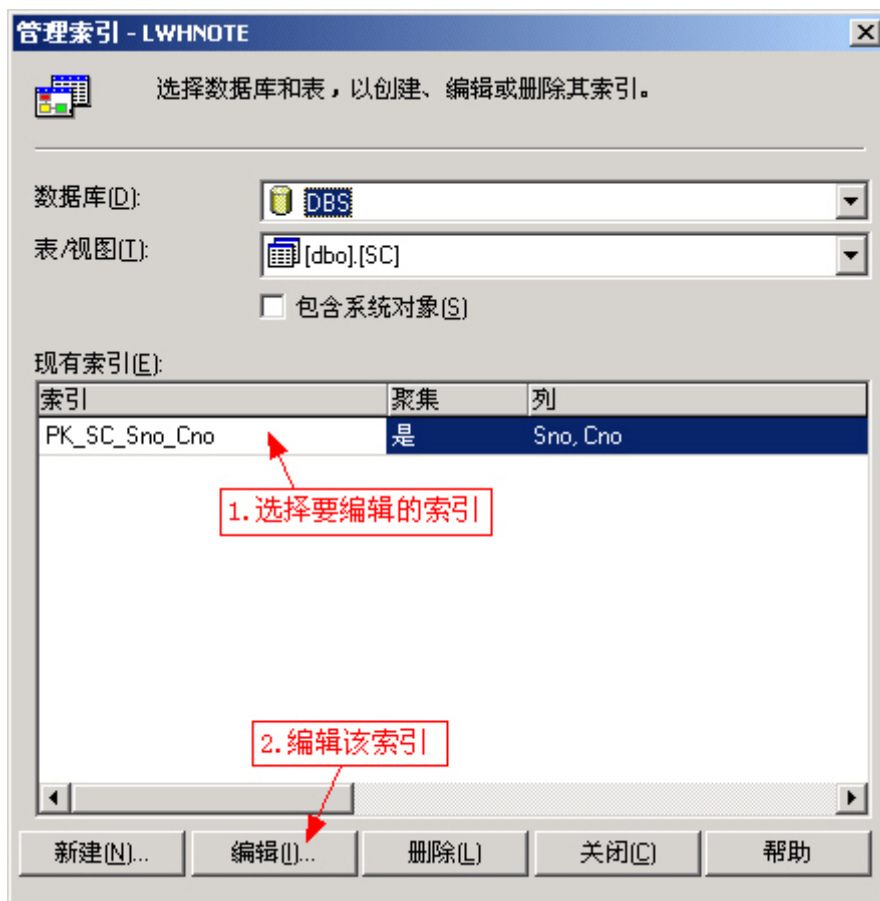


在索引属性对话框中，建立新的索引

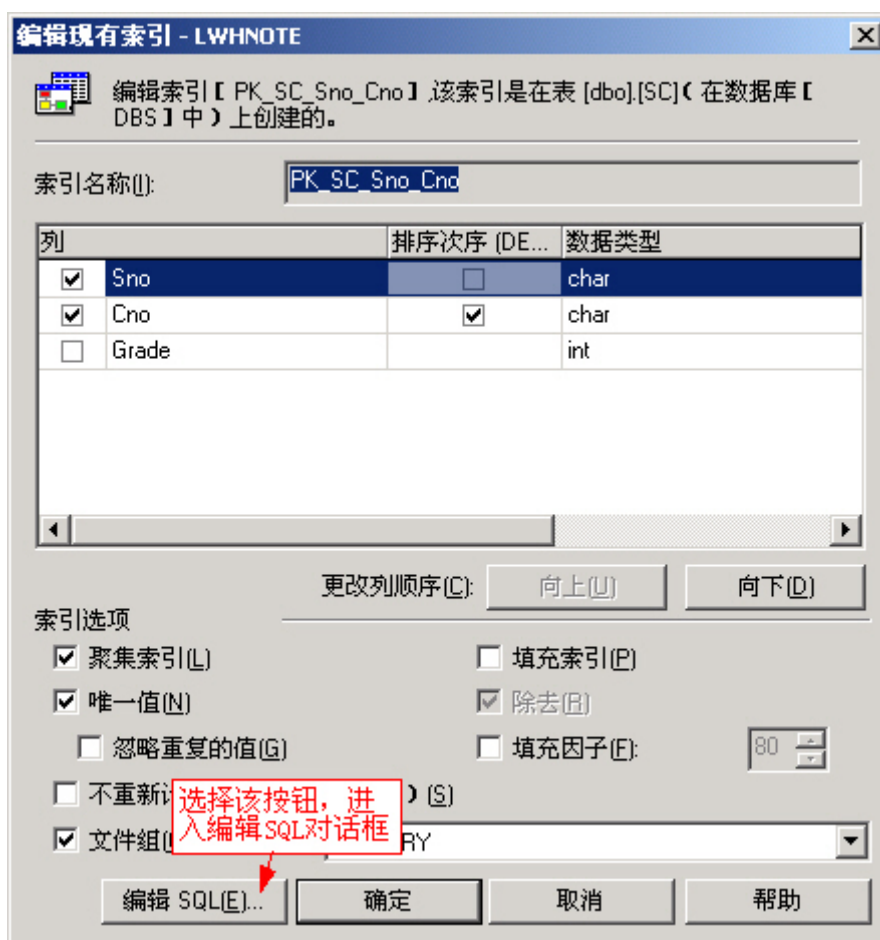


二、用查询分析器创建索引

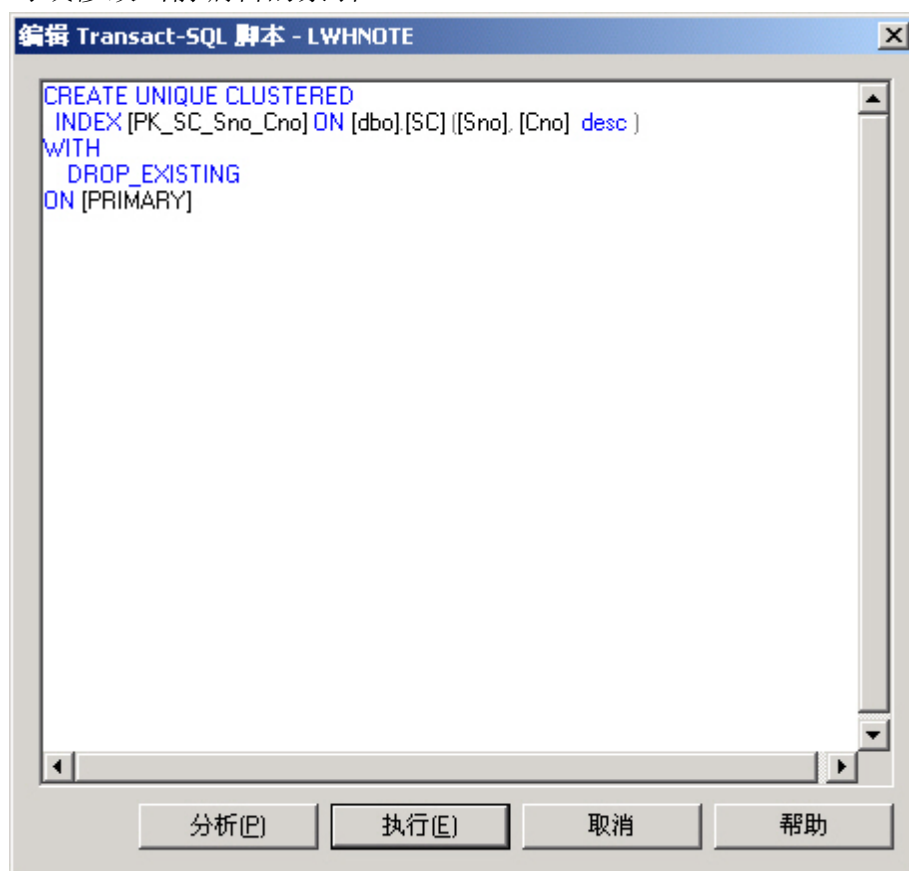
例 1: 对学生数据表 Student 中的姓名 Sname 建立一个索引。



在下面的对话框中，修改索引

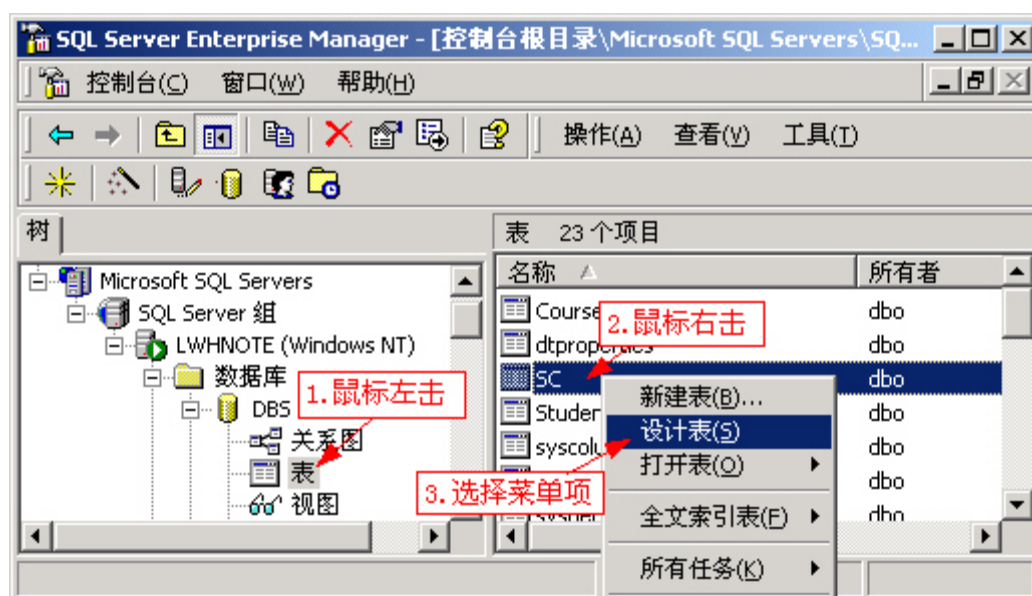


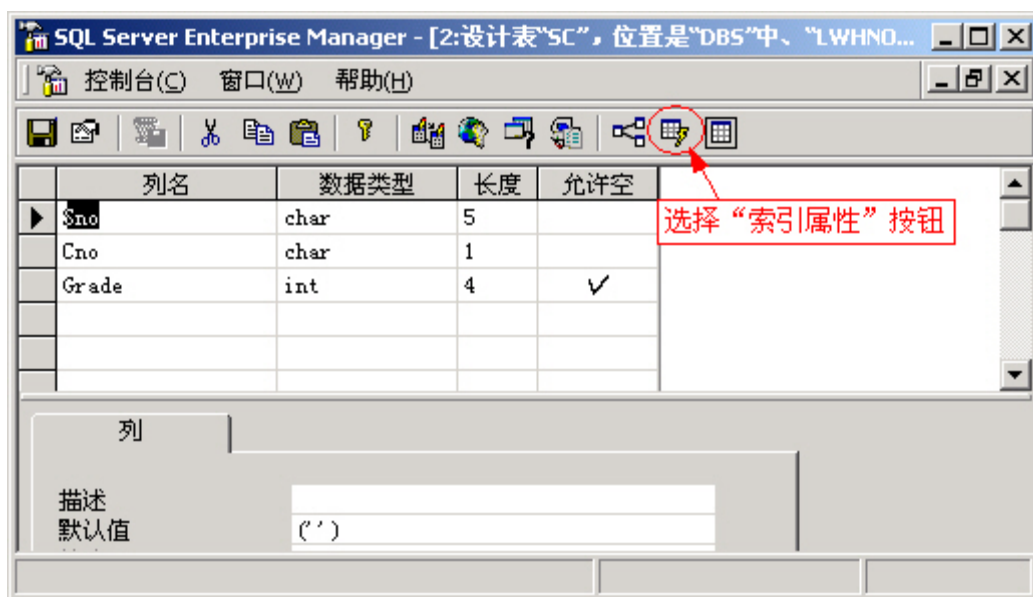
此外，还可以在上面的对话框中，按“编辑 SQL”按钮，进入当前 SQL 脚本编辑框，便于对照学习或修改当前编辑的索引。



B. 用表的属性对话框修改索引

例：对在 2.2.1 中创建索引 PK_SC_Sno_Cno 进行修改。





在上面的对话框中，按下“索引属性”按钮，弹出如下索引编辑框，可以对要修改的索引进行编辑。



二、用查询分析器修改索引

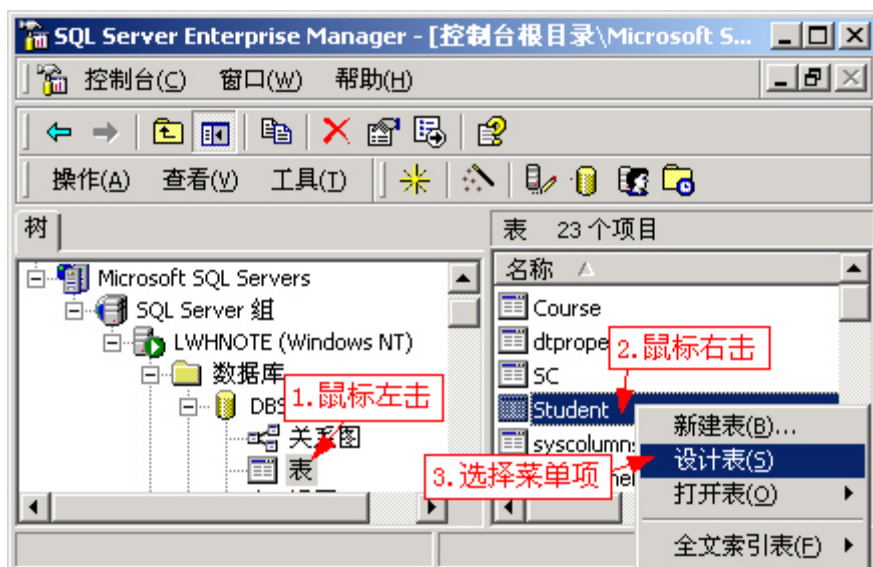
在查询分析器中，一般没有必要用查询分析器修改索引。若要修改一个索引，常常删除该索引，然后再创建索引。

2.2.3 删除索引

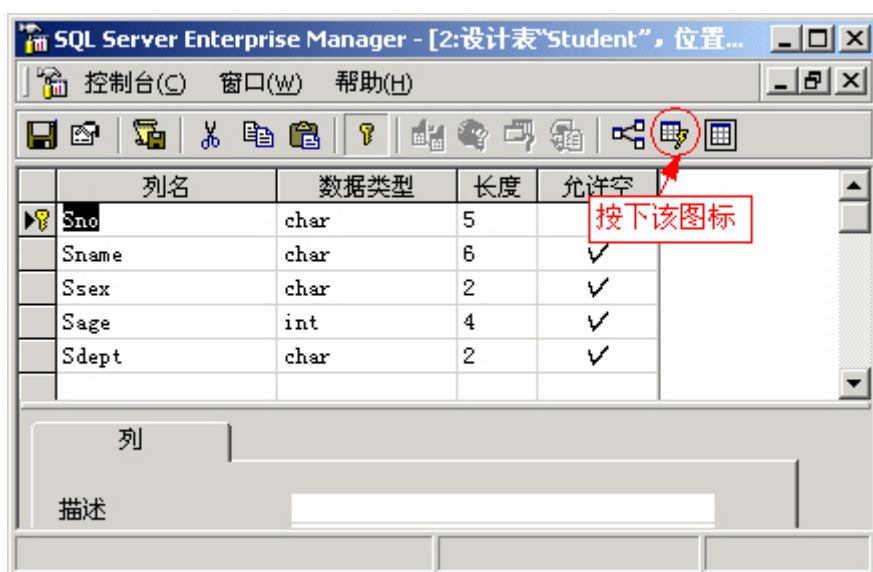
 帮助索引：<删除索引|删除索引>

一、用企业管理器修改索引

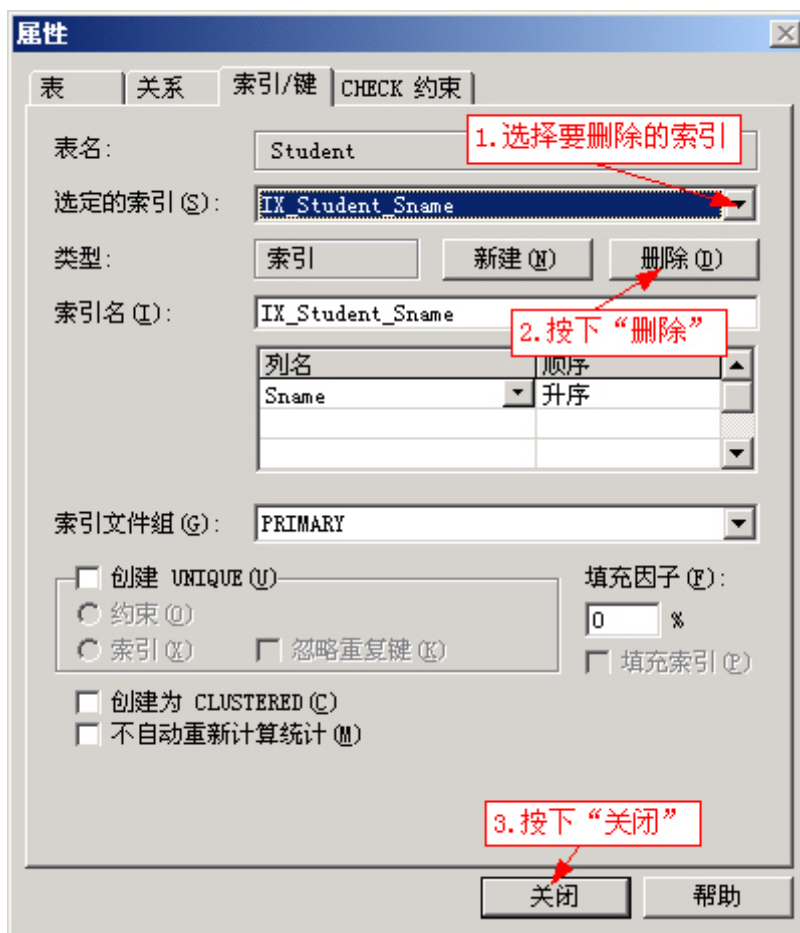
例：删除学生数据表 Student 中的索引 IX_Student_Sname。



在下面对话框中，按下索引属性按钮 

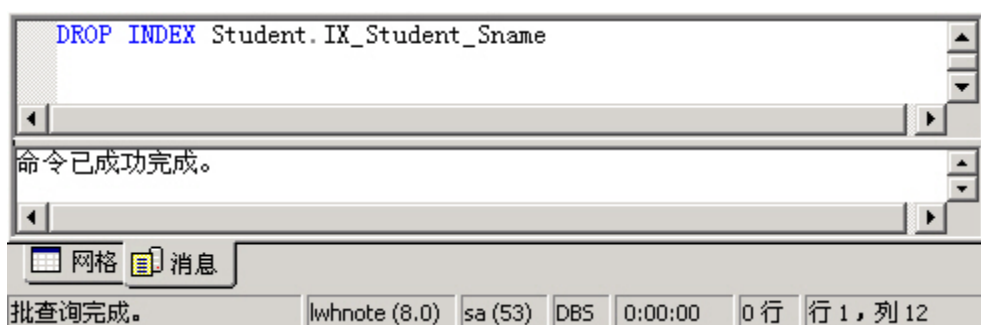


弹出如下对话框，选择要修改的索引



二、用查询分析器删除索引

例：删除学生数据表 Student 中的索引 IX_Student_Sname。

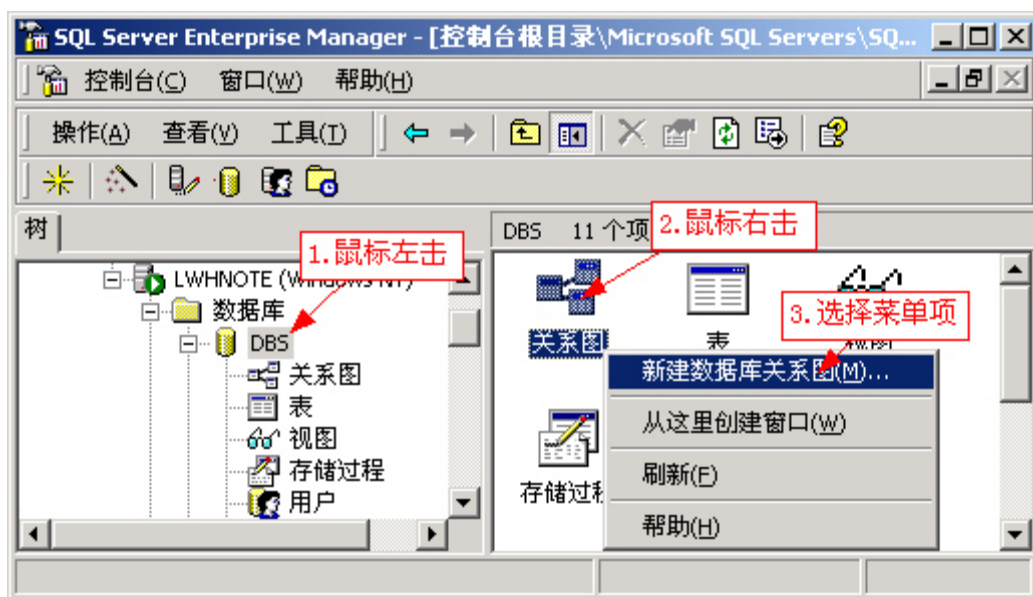


2.2.4 创建和使用关系图

 帮助索引：<数据库关系图|表视图>

一、用企业管理器创建关系图

例：创建学生成绩管理系统 DBS 的一个关系图。



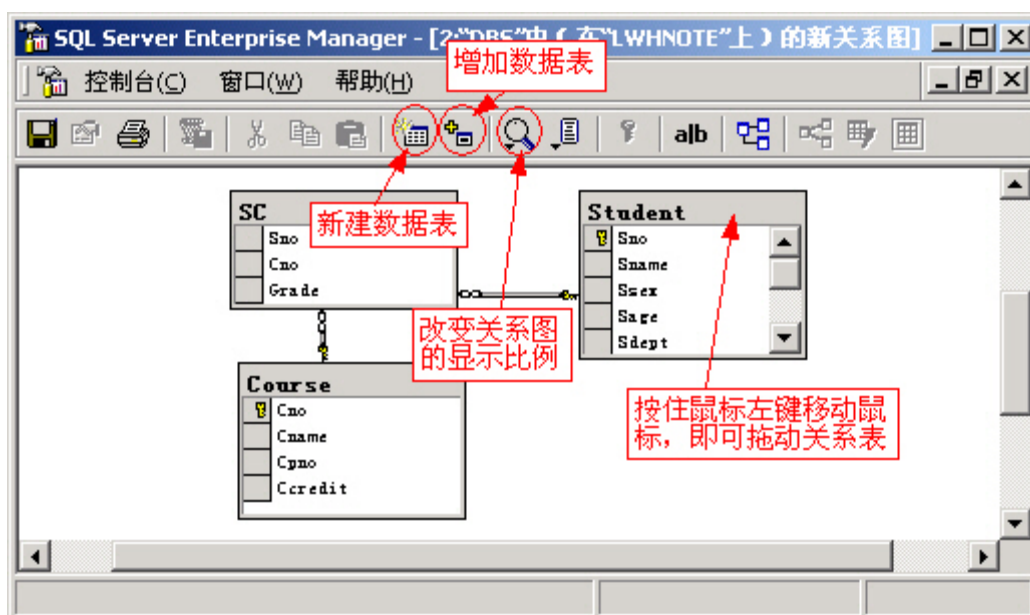
在下面对话框中，从左边的可用表中选择 DBS 的三个表到右边列表框



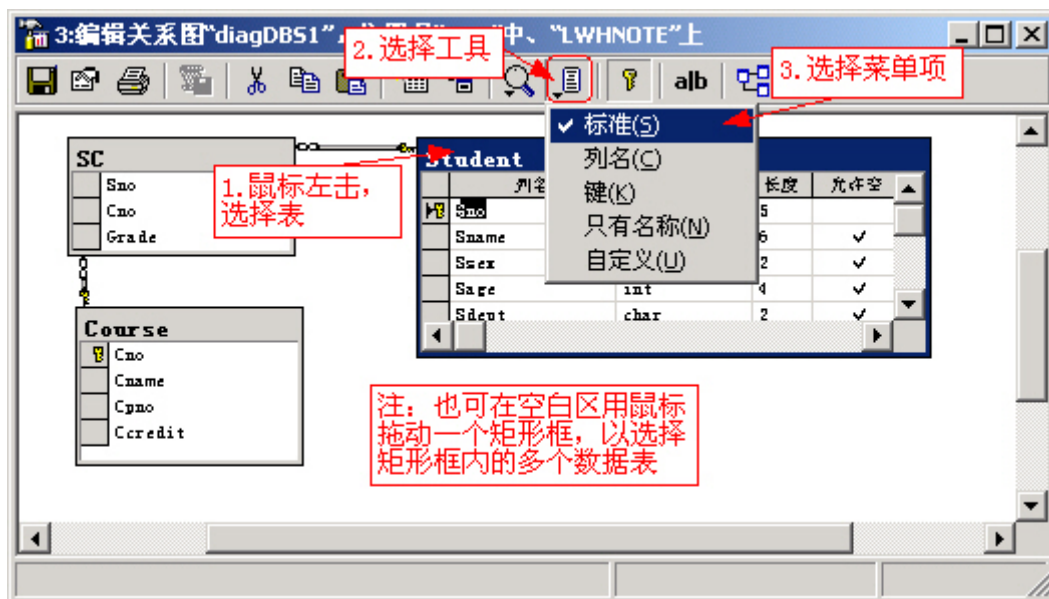
在下面的对话框中，按“完成”按钮，即可创建 DBS 的数据库关系图



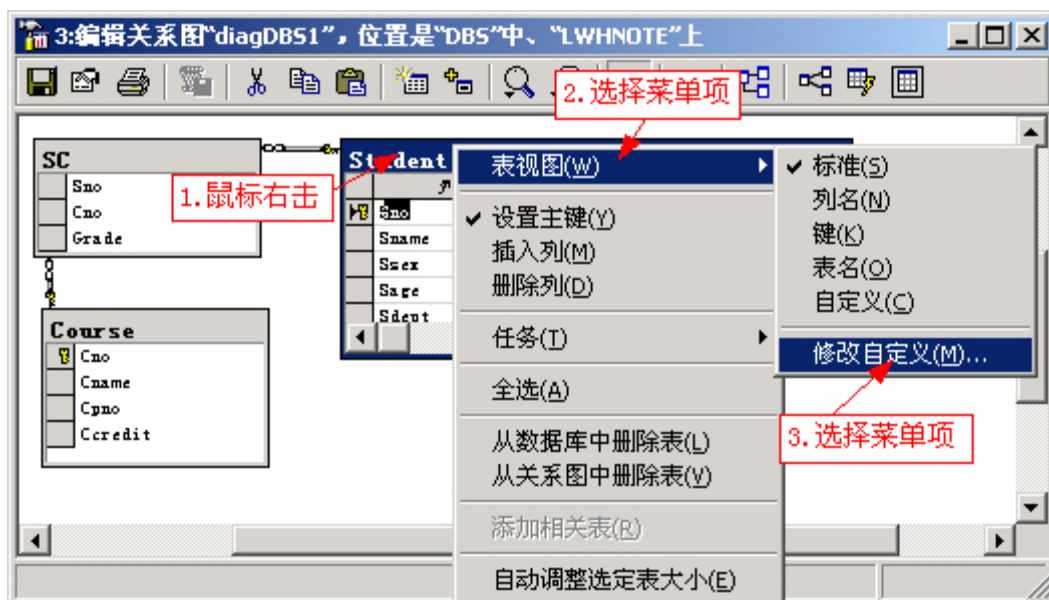
下面的画面，即可刚才创建的 DBS 关系图。在下图中，可以用鼠标拖动关系图中各个关系表的位置、可以在关系图中建立新表、增加已经建立的关系表、改变关系图的显示比例。



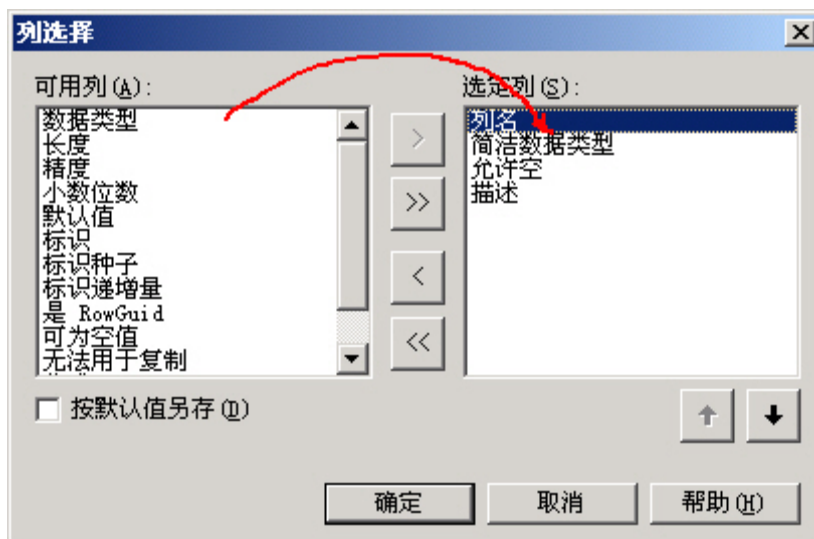
用下图所示的方法，可以改变关系表的显示内容



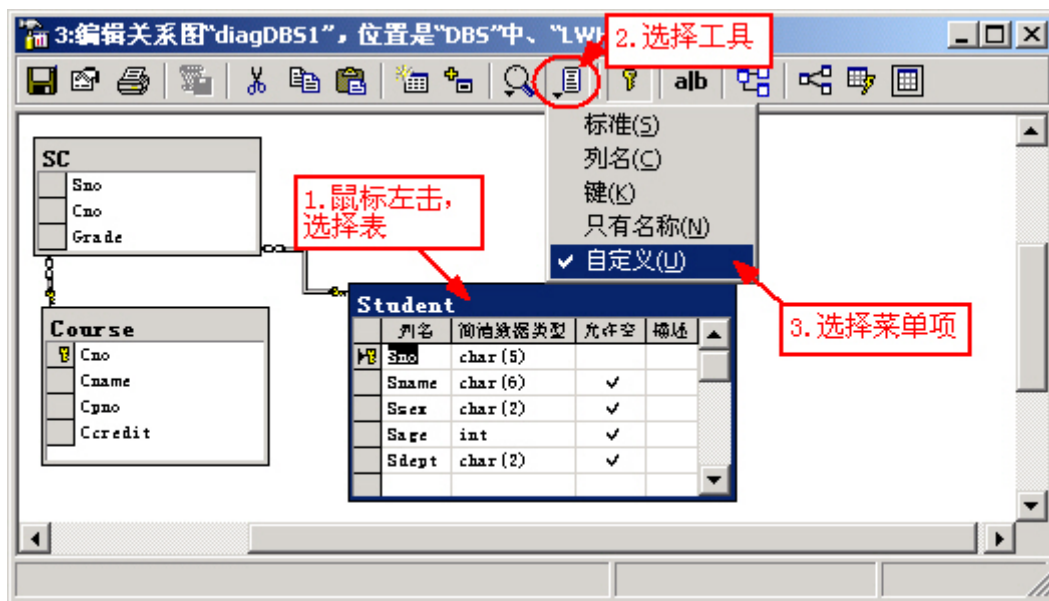
还可以通过下面三个图所示的方法, 自己定义显示样式。下图为弹出自定义显示样式菜单



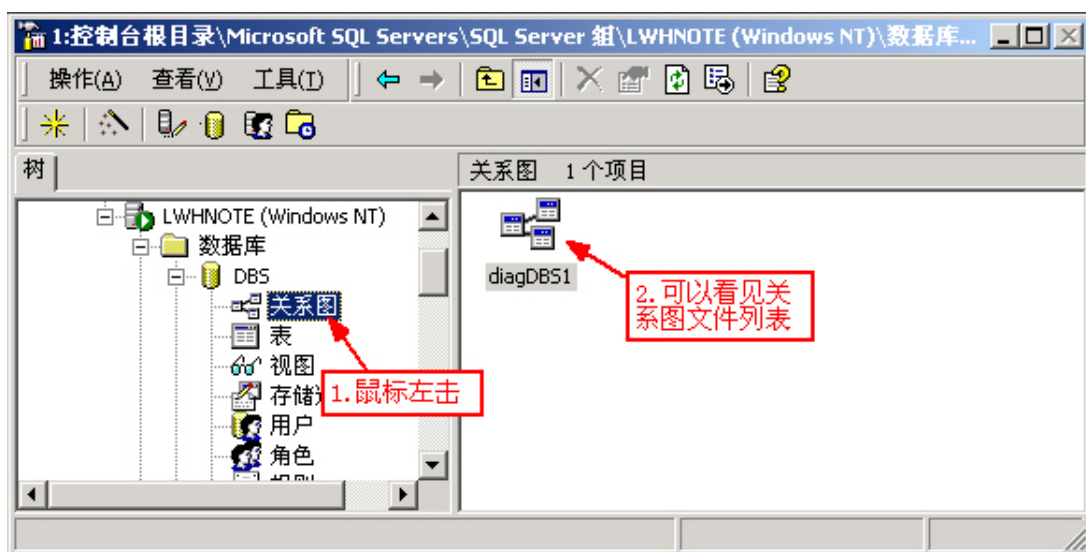
下图定制自定义显示样式



然后，对指定关系表（如：Student）使用自定义显示样式

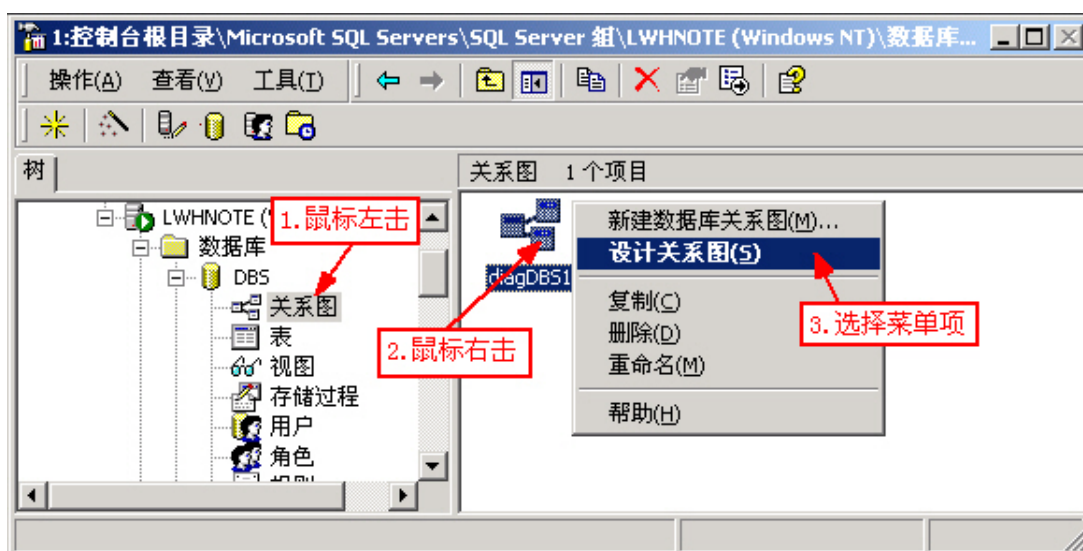


最后，按保存 ，必要时输入关系图的名称（如：diagDBS1）可以将关系图保存，退出后，在企业管理器中可以看到如下图所示

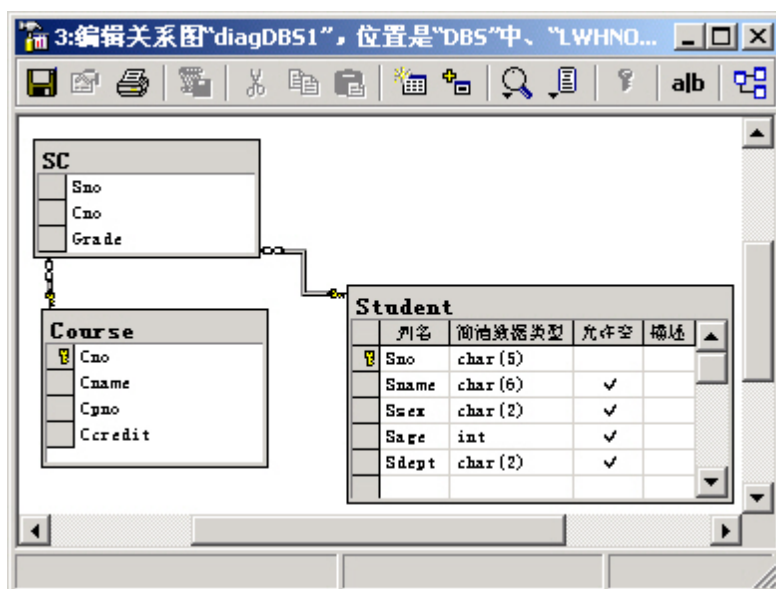


二、用企业管理器修改关系图

例：修改学生成绩管理系统 DBS 中的关系图 diagDBS1。



下面显示了关系图 **diagDBS1**，可以对其进行各种修改，包括：增加数据表、修改关系图中各个表的排列位置、改变一个或多个数据表的显示样式、自定义数据表的显示样式等。



2.3 实验内容

阅读本实验 2.1~2.2 的内容，完成下面的实验内容。

一、恢复数据库

用备份文件 DBS_ALL 恢复数据库 DBS，具体操作参见[<1.7.3 数据库恢复>](#)。

二、创建数据库表

① 使用企业管理器在数据库 myDB（若不存在，创建一个 myDB）中创建一个数据表

TestTab，数据表中包括各种常用数据类型：**INT**、**REAL**、**NUMERIC**、**VARBINARY**、**BIT**、**CHAR**、**VARCHAR**、**TEXT**、**IMAGE**、**DATETIME**、**MONEY**，列名可以自己定义，如 **Col1**、**Col2**，…。然后向这个数据表中追加一些记录，并注意 **BIT**、**DATETIME** 等数据类型数据的输入。

② 用企业管理器修改上面的数据表 **TestTab**，删除数据类型为 **NUMERIC**、**VARBINARY**、**IMAGE** 数据列，并使数据类型为 **INT** 的数据列为主关键字列，数据类型为 **REAL** 的数据列不能为空、缺省值为 **1**，数据类型为 **MONEY** 的数据列的值只能是 **1~1000** 之间的值，数据类型为 **CHAR** 的数据列必须唯一。

③ 使用查询分析器创建一个学生成绩管理系统DBS中的各个数据表**Student**、**Course**、**SC**，其数据表的结构及各种约束关系参见 [2.1.2](#) 中<二、用查询分析器创建数据表>中例 2。保存创建数据表的语句为一个Sql文件，如：**CreateTabs.sql**。

三、更新数据库表中数据

① 使用查询分析器向数据库DBS的各个数据表中追加一些数据记录，其记录内容参见 [2.1.5](#) 中<一、插入数据>< B. 用查询分析器插入数据><例 1~3>。

② 使用查询分析器修改数据表中的数据，如：修改学生表 **Student** 中的系号 **Sdept** 均为大写（字符串转换为大写的函数为 **UPPER()**），如：删除成绩表中所有不及格的记录。

四、建立数据库的索引

① 分别使用企业管理器和查询分析器给数据库 **DBS** 中的各个数据表建立关键字、外键，给 **Student** 数据表建立一个 **Sname** 的降序索引，给 **SC** 数据表建立一个 **Grade** 的降序索引。

② 使用企业管理器建立数据库 **DBS** 的关系图，试着改变关系图布局、各个关系表的显示样式。

五、为下次实验准备

① 建立学生成绩管理系统 **DBS** 并备份到文件 **DBS_ALL** 中，并用软盘或 U 盘保存。

② 将建立 **DBS** 数据库的所有表的过程存为一个 **SQL** 文件 **CreateTabs.sql**，并用软盘或 U 盘保存。

③ 将追加 DBS 数据库各个表的过程分别存为一个 SQL 文件：InsStudent.sql、InsCourse.sql、InsSC.sql，并用软盘或 U 盘保存。

2.4 思考题

① 如何给日期型数据列设置默认值？标识符属性列和全局唯一标识符属性列如何定义？这些列有什么用处？

② 掌握数据列的以下主要数据类型：INT、REAL、NUMERIC、VARBINARY、BIT、CHAR、VARCHAR、TEXT、IMAGE、DATETIME、MONEY。

③ 如何查看用企业管理器定义的数据表的 SQL 语句？

④ 如何将查询分析器中内容保存到一个 SQL 文件？其作用是什么？

⑤ 如何在查询分析器中打开、运行一个已有的 SQL 文件？

⑥ 如何检验各种约束是否起作用？

⑦ 如何使用 SELECT 语句一次输入多条记录？

⑧ 在删除数据表时，你试过有外关键字约束的数据表吗？有什么要值得注意？

⑨ 如何自定义数据库关系图中数据表的显示样式？

⑩ 能够理解建立索引的必要性吗？是否能设计一个方案来验证建立索引的必要性？

(11) 数据库系统是在什么时候、如何使用索引？