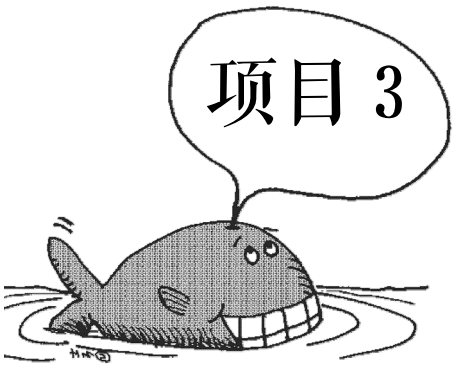

创建和管理表

表是数据库中最重要的基础对象，它包含数据库中的所有数据，其他数据库对象（例如索引和视图等）都是依赖于表而存在的。若要使用数据库来存储和组织数据，首先就需要创建表。在本项目中将通过 15 个任务来创建和管理表，主要内容包括表的设计、理解 SQL Server 2008 中的数据类型，以及创建和修改表等。

任务 1　表的设计

任务描述

在本任务中，设计一个用于管理学生成绩的数据库，要求使用 Office Visio 软件画出如图 3.1 所示的数据库模型图。

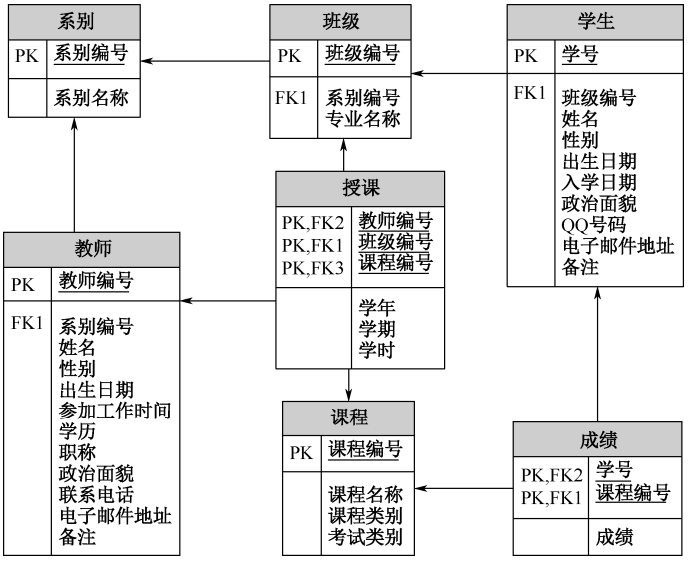

图 3.1　数据库模型图

任务实现

实现步骤如下：

（1）对相关部门进行调查，在此基础上进行需求分析，确定“学生成绩”数据库中包含 7 个表，分别用于存储系别、班级、学生、教师、授课、课程和成绩信息。

（2）确定每个表中包含的哪些列、每个列是何种数据类型以及在哪个列建立主键。

（3）使用 Office Visio 软件绘制出如图 3.1 所示的数据库模型图。

在如图 3.1 所示的数据库模型图中，一共包含 7 个实体，分别是系别、班级、学生、教师、授课、课程和成绩，每个实体对于一个表。每个实体都包含一组属性，每个属性对应于表中的一列。在数据库模型图中，每个实体用一个表格来表示，表格的第一行列出表的名称，第二行列出表的主键列及其数据类型（主键也可以由多个列组成，从而占用多行），其他各行分别列出剩余各列的名称。根据需要也可以在数据库模型图包含各列的数据类型。

在数据库模型图中，PK 表示相应列是表的主键。例如，“学号”列是“学生”表中的主键，“学号”和“课程编号”列组成了“成绩”表中的主键，“教师编号”、“班级编号”和“课程编号”3 个列共同组成了“授课”表的主键。FK 表示相应列是表中的外键。例如，在“学生”表中将“班级编号”列标记为 FK1，该列是学生表中的外键，它指向“班级”表中的主键，亦即“班级编号”列；在“成绩”表中，“学号”和“课程编号”列组成了主键，这两个列同时又是外键，它们分别指向“学生”表和“成绩”表中的候选键。

如果两个表之间存在着关系，则它们会通过一个带有箭头的线段连接起来，箭头指向的表称为父表（即主键所在的表），线段另一端所连接的表称为子表（即外键所在的表）。

在本书后面的项目和任务中，将按照如图 3.1 所示的数据库模型图来实现一个用于学生成绩管理的数据库，然后在该数据库中创建各个表以及其他数据库对象。

相关知识

数据库设计是数据库应用程序开发的一个重要环节，数据库设计的核心就是表的设计。表是包含 SQL Server 数据库中的所有数据的对象。表是由一些列组成的集合。数据在表中的组织方式与电子表格相似，都是按行和列的格式组织的，行和列是表的两个主要组件。此外，通常还需要在表中创建各种约束，以确保数据的完整性。

一、制订表规划

设计数据库时，必须先确定数据库所需要的所有表、每个表中数据的类型以及可访问每个表的用户。合理的表结构可提高数据查询效率。如果在实现表之后再做大的修改，将会耗费大量的时间。在创建表及其对象之前，最好先制订出规划并确定表的下列特征。

1. *表要存储什么对象*

由表建模的对象可以是一个有形实体，例如一个人或一个产品；也可以是一个无形项目，例如某项业务、公司中的某个部门。通常会有少数主对象，标识它们后将显示相关项。主对象及其相关项分别对应于数据库中的各个表。表中每一行是一条唯一的记录，代表由表建模的对象的一个单独实例。每一列代表记录中的一个字段，代表由表建模的对象的某个属性。

2. *表中每一列的数据类型和长度*

由表建模的对象的各个属性分别通过不同的列来存储，设计表时应确定每一列使用何种数

据类型。例如，姓名列应该使用字符串类型；出生日期列应使用日期时间数据类型；成绩列应使用数字数据类型，可以带 1 位小数或不带小数。

3. 表中哪些列允许空值

在数据库中，空值用 NULL 来表示，这是一个特殊值，表示未知值的概念。NULL 不同于空字符或 0。实际上，空字符是一个有效的字符，0 是一个有效的数字。NULL 仅表示此值未知这一事实。使用 NOT NULL 约束可以指定列不接受空值。

4. 是否要使用以及在何处使用约束、默认值和规则

约束定义关于列中允许值的规则，是强制实施完整性的标准机制。约束分为列级约束和表级约束，前者应用于单列，后者应用于多列。SQL Server 2008 支持下列约束。

（1）CHECK 约束：定义列中哪些数据值是可接受的。可以将 CHECK 约束应用于多列，也可以为一列应用多个 CHECK 约束。当删除表时，将同时删除 CHECK 约束。

（2）DEFAULT 约束：为表列定义的属性，指定要用作该列的默认值的常量。如果插入或更新数据时为该列指定了 NULL 值，或者没有为该列指定值，则会把在 DEFAULT 约束中定义的常量值放置在该列中。

（3）UNIQUE 约束：基于非主键强制实体完整性的约束。UNIQUE 约束可以确保不输入重复的值，并确保创建索引来增强性能。

默认值和规则都是数据库对象，默认值是在用户未指定值的情况下系统自动分配的数据值，规则是绑定到列或用户定义数据类型并指定列中可接受哪些数据值的数据库对象。

5. 使用何种索引以及在何处使用索引

索引是关系数据库中的一种对象，它可以基于键值快速访问表中的数据，也可以强制表中行的唯一性。SQL Server 支持聚集索引和非聚集索引。在全文搜索中，全文索引存储有关重要词及其在给定列中位置的信息。设计表时，应考虑在哪些列上使用索引，是使用聚集索引、非聚集索引还是全文索引。

6. 哪些列是主键或外键

主键（PK）是表中的一列或一组列，可以用来唯一地标识表中的行。外键（FK）是表中的一列或列组合，其值与同一个表或另一个表中的主键或唯一键相匹配，也称为引用键。主键和外键分别通过 PRIMARY KEY 和 FOREIGN KEY 约束来实现。

（1）PRIMARY KEY 约束：标识具有唯一标识表中行的值的列或列集。在一个表中，不能有两行具有相同的主键值。不能为主键中的任何列输入 NULL 值。建议使用一个小的整数列作为主键。每个表都应有一个主键，表的主键将自动创建索引。限定为主键值的列或列组合称为候选键。尽管不要求表必须有主键，但最好定义主键。

（2）FOREIGN KEY 约束：标识并强制实施表之间的关系。一个表的外键指向另一个表的候选键。

创建表的最有效的方法是同时定义表中所需要的所有内容，这些内容包括表的数据限制和其他组件。在创建和操作表后，将对表进行更为细致的设计。

创建表的有用方法是：创建一个基表并向其中添加一些数据，然后使用这个基表一段时间。这种方法可以在添加各种约束、索引、默认设置、规则和其他对象形成最终设计之前，发现哪些事务最常用，哪些数据经常输入。

完成所有表的设计后，可以使用 Microsoft Office Visio 将设计结果绘制成一张数据库模型图，用来描述数据库的结构，表示数据库中包含哪些表，每个表中包含哪些列，每个列使用什

么数据类型，哪些表之间通过主键和外键约束建立了关系。

二、规范化逻辑设计

数据库的逻辑设计（包括各种表和表间关系）是优化关系型数据库的核心。设计好逻辑数据库，可以为优化数据库和应用程序性能打下基础。逻辑数据库设计不好，会影响整个系统的性能。

规范化逻辑数据库设计包括使用正规的方法来将数据分为多个相关的表。拥有几个具有较少列的窄表是规范化数据库的特征，而拥有少量具有较多列的宽表是非规范化数据库的特征。一般而言，合理的规范化会提高性能。如果包含有用的索引，则 SQL Server 查询优化器可以有效地在表之间选择快速、有效的连接。

规范化具有以下好处：使排序和创建索引更加迅速；聚集索引的数目更大；索引更窄、更紧凑；每个表的索引更少，这样将提高 INSERT、UPDATE 和 DELETE 语句的性能；空值更少，出现不一致的机会更少，从而增加数据库的紧凑性。

随着规范化的不断提高，检索数据所需的连接数和复杂性也将不断增加。表间的关系连接太多、太复杂可能会影响性能。通常很少包括经常性执行且所用连接涉及 4 个以上表的查询。

在某些情况下，逻辑数据库设计已经固定，全部进行重新设计是不现实的。但是，尽管如此，将大表有选择性地进行规范化处理，分为几个更小的表还是可能的。如果通过存储过程对数据库进行访问，则在不影响应用程序的情况下架构可能发生更改。如果不是这种情况，那么可以创建一个视图，以便向应用程序隐藏架构的更改。

在关系数据库设计理论中，规范化规则指出了在设计良好的数据库中必须出现或不出现的某些属性。关于规范化规则的完整讨论超出了本书的范畴。下面仅给出获得合理的数据库设计的一些规则。

（1）表应该有一个标识符。数据库设计理论的基本原理是：每个表都应有一个唯一的行标识符，可以使用列或列集将任何单个记录与表中的所有其他记录区别开来。每个表都应有一个 ID 列，任何两个记录都不能共享与一 ID 值。作为表的唯一行标识符的列是表的主键，例如，在“学生”表中可使用“学号”列作为表的主键，但很少使用“姓名”列作为表的主键，因为在同名同姓的现象在一个学校甚至一个班都是屡见不鲜的。

（2）表应只存储单一类型实体的数据。试图在表中存储过多的信息会影响对表中的数据进行有效、可靠的管理。在 AdventureWorks 示例数据库中，销售订单和客户信息存储在不同的表中。虽然可以在一个表中创建包含有关销售订单和客户信息的列，但是这样设计会导致出现一些问题。必须在每个销售订单中另外添加和存储客户信息、客户姓名和地址，这将使用数据库中的其他存储空间。如果客户地址发生变化，必须更改每个销售订单。另外，如果从 Sales.SalesOrderHeader 表中删除了客户最近的销售订单，则该客户的信息将会丢失。

（3）表应避免可为空的列。表中的列可定义为允许空值。虽然在某些情况下，允许空值可能是有用的，但是应尽量少用。这是因为需要对它们进行特殊处理，从而会增加数据操作的复杂性。如果某一表中有几个可为空值的列，并且列中有几行包含空值，则应考虑将这些列置于链接到主表的另一表中。通过将数据存储在两个不同的表中，主表的设计会非常简单，而且仍能够满足存储此信息的临时需要。

（4）表不应有重复的值或列。数据库中某一项目的表不应包含有关特定信息的一些值。例

如，AdventureWorks 示例数据库中的某产品可能是从多个供应商处购买的，如果 Production.Product 表有一列为供应商名称，就会产生问题。一个解决方案是将所有供应商名称存储在该列中。但是，这使得列出各个供应商变得非常困难。另一个解决方案是更改表的结构来为另一个供应商名称再添加一列。但是，这只允许有两个供应商。此外，如果一个产品有 3 个供应商，则必须再添加一列。如果发现需要在单个列中存储多个值，或者一类数据（例如 PhoneNumber1 和 PhoneNumber2）对应于多列，则应考虑将重复数据置于链接到主表的另一个表中。例如，在 AdventureWorks 示例数据库中，Production.Product 表用于存储产品信息，Purchasing.Vendor 表用于存储供应商信息，此外还有第三个表 Purchasing.ProductVendor。第三个表只存储产品的 ID 值和产品供应商的 ID 值。这种设计允许产品有任意多个供应商，既不需要修改表的定义，也不需要为单个供应商的产品分配未使用的存储空间。

任务 2　认识 SQL Server 数据类型

任务描述

在本任务中，首先创建一个名为“学生成绩”的数据库，然后在该数据库中创建 4 个用户定义数据类型，如图 3.2 所示。

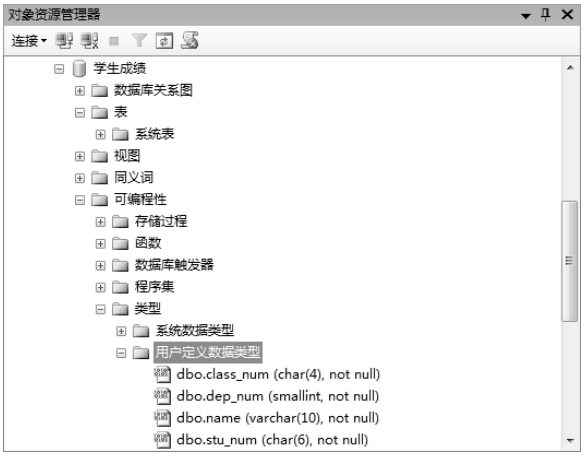

图 3.2　在数据库中创建用户定义数据类型

任务实现

一、使用 SQL Server Management Studio 创建用户定义数据类型

操作要求：创建一个名为“学生成绩”的数据库，然后使用 SQL Server Management Studio 在该数据库中创建一个名为 dep_num 的用户定义数据类型（可用于定义“系别编号”列），该数据类型基于系统数据类型 smallint，不允许取空值。

实现步骤如下：

（1）启动 SQL Server Management Studio，连接到 SQL Server 数据库引擎。

（2）创建一个名为“学生成绩”的数据库。

（3）展开“ 学生成绩 ”数据库，依次展开“ 可编程性 ”和“ 类型 ”，右键单击“ 用户定义数据类型 ”，然后选择“ 新建用户定义数据类型 ”，如图所示 3.2 所示。

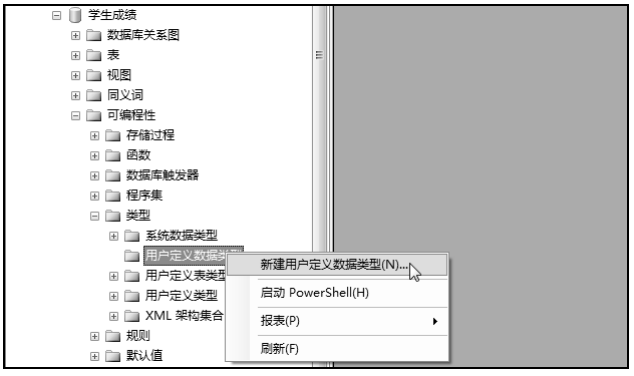

图 3.2　创建用户定义数据类型

（4）当出现如图 3.3 所示的“ 新建用户定义数据类型 ”对话框时，从包含当前用户的所有可用架构的“ 架构 ”列表中选择一个架构，默认选择是当前用户的默认架构（例如 dbo）。

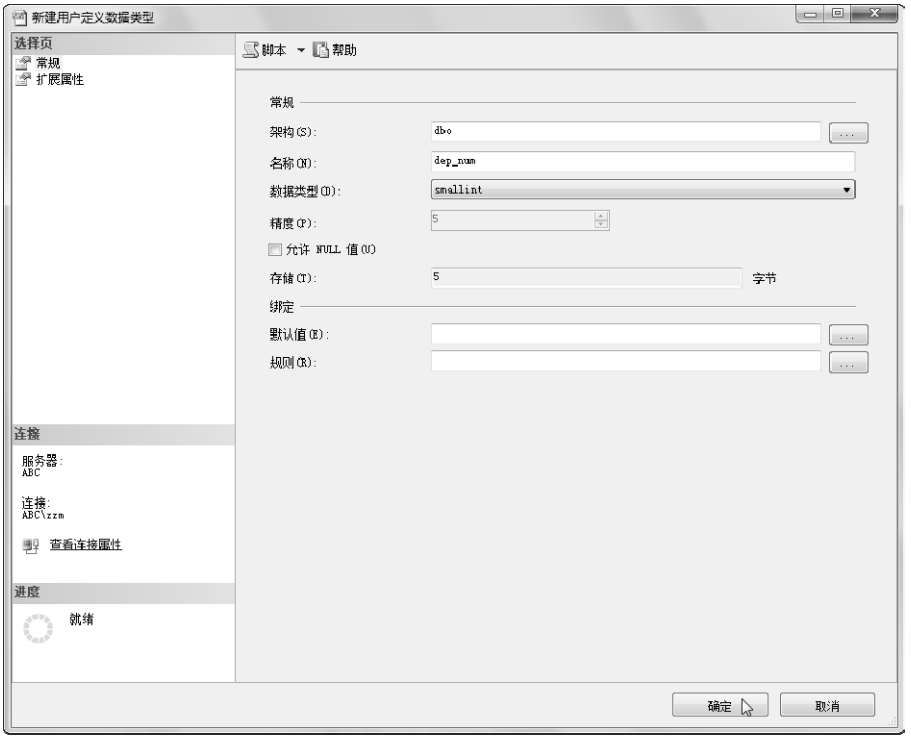

图 3.3 “ 新建用户定义数据类型 ”对话框

（5）在“ 名称 ”框中输入用于在整个数据库中标识 UDDT 的唯一名称，在本任务中数据类型名称为 dep_num。该名称的最大字符数必须符合 sysname 系统数据类型的要求。

（6）从“ 数据类型 ”下拉列表框中选择一种基本数据类型，在本任务中选择了 smallint 类型。该列表框列出了除 timestamp 数据类型之外的所有数据类型。

（7）按需要选取或不选取“ 允许空值 ”复选框，以指定 UDDT 是否可以接受 NULL 值。

在本任务中未选取该复选框，意即不允许空值。

（9）单击“确定”按钮，完成 UDDT 的创建。

二、使用 CREATE TYPE 语句创建用户定义数据类型

操作要求：使用 CREATE TYPE 语句创建以下用户定义数据类型。

（1）class_num：基于 char 类型，长度为 4，非空，可用于定义班级编号列。

（2）stu_num：基于 char 类型，长度为 6，非空，可用于定义学号列。

（3）name：基于 varchar 类型，长度为 10，非空，可用于定义姓名列。

实现步骤如下：

（1）启动 SQL Server Management Studio，连接到 SQL Server 数据库引擎。

（2）创建一个新查询，然后在查询编辑窗口中输入以下语句。

```
USE 学生成绩;
GO
CREATE TYPE class_num FROM char(4) NOT NULL;
GO
CREATE TYPE stu_num FROM char(6) NOT NULL;
GO
CREATE TYPE name FROM varchar(10) NOT NULL;
GO
```

（3）将脚本文件保存为 SQLQuery3-01.sql，按 F5 键执行 SQL 脚本。

相关知识

设计表时首先要执行的一个操作就是为表中的每个列指定数据类型。数据类型定义了各列允许使用的数据值。例如，如果希望列中只含有名称，则可以将一种字符数据类型指定给列；如果希望列中只包含数字，则可以指定一种数字型数据类型。

一、数据类型概述

数据类型是一种属性，用于指定对象可保存的数据的类型：整数数据、字符数据、货币数据、日期和时间数据以及二进制字符串等。在 SQL Server 中，每个列、局部变量、表达式和参数都具有一个相关的数据类型。

SQL Server 2008 中的数据类型可以归纳为数字数据、字符串、日期和时间以及其他数据类型 4 个类别。

二、数字数据类型

属于数字数据类型的数字可以参与各种数学运算。数字数据类型可以分为整数类型和小数类型；小数类型又分为近似数字类型和精确数字类型。

1. 整数类型

整数类型包括 bigint、int、smallint、tinyint 和 bit。

bigint 数据类型用于存储从-2^{63}（−9 223 372 036 854 775 808）到$2^{63}-1$（9 223 372 036 854 775 807）的整数，占用 8 个字节。

bit 数据类型用于存储整数，但只能取 0、1 或 NULL。若表中的列为 8 bit 或更少，则这些列作为 1 个字节存储。若列为 9 到 16 bit，则这些列作为 2 个字节存储，以此类推。字符串值 TRUE 和 FALSE 可以转换为以下 bit 值：TRUE 转换为 1，FALSE 转换为 0。

int 数据类型用于存储从-2^{31}（−2 147 483 648）到$2^{31}-1$（2 147 483 647）的整数，占用 4 个字节。

smallint 数据类型用于存储从-2^{15}（−32 768）到$2^{15}-1$（32 767）的整数，占 2 个字节。

tinyint 数据用于存储从 0 到 255 的整数，占用 1 个字节。

2. 小数类型

小数类型包括近似数字类型 float(*n*)和 real、精确数字类型 decimal(p, s)和 numeric(p, s)以及货币数字类型 money 和 smallmoney。

float(*n*) 数据类型用于存储从−1.79E+308 到 1.79E+308 之间的浮点数。*n* 为用于存储科学记数法浮点数尾数的位数，同时指示其精度和存储大小。*n* 必须为从 1 到 53 之间的值。当 *n* 介于 1 和 24 之间时，精度为 7 位有效数字，占用 4 个字节；当 *n* 介于 25 和 53 之间时，精度为 15 位有效数字，占用 8 个字节。

real 数据类型相当于 float(24)，用于存储−3.40E−38～3.40E−38 之间的浮点数，占用 4 个字节。

decimal(p, s) 和 numeric(p, s) 数据类型用于存储带小数点且数值确定的数据，可把它们视为相同的数据类型。p 表示数值的全部位数，取值范围为 1～38，其中包含小数部分的位数，但不包括小数点在内，p 值称为该数值的精度（precision）；s 表示小数的位数（scale）。整数部分的位数等于 p 减去 s。例如，decimal(10, 5)表示数值中共有 5 位整数，其余 5 位是小数部分，该列的精度为 10 位。将一个列指定为 decimal 类型时，如果没有指定精度 p，则默认精度为 18 位；如果没有指定小数位数，则默认小数位数为 0。

在 decimal 和 numeric 数据类型中，p 不仅表示数值精度，也指定了数据占用存储空间的大小。当 p 介于 1 和 9 之间时，数据占用 5 个字节；当 p 介于 10 和 19 之间时，数据占用 9 个字节；当 p 介于 20 和 28 之间时，数据占用 13 个字节；当 p 介于 29 和 38 之间时，数据占用 17 个字节。

money 和 smallmoney 数据类型用于存储货币数据，这些类型的数据实际上都是带有 4 位小数的 decimal 类型数据。在 money 或 smallmoney 类型的列中输入货币数据时，必须在数值前面加上一个货币符号。输入负值时，应当在货币数据后面加一个负号。在输入过程中，不需要每 3 位数字加一个逗号来分隔，但在打印过程中系统会自动添加逗号。在数据库中创建表时，如果将一个列指定为 money 数据类型，则这个列的取值范围为-2^{63}（−922 337 203 685 477.5808）～$2^{63}-1$（+922 337 203 685 477.5807），这样一个列在内存中占用 8 个字节，前面 4 个字节表示货币值的整数部分，后面 4 个字节表示货币值的小数部分。smallmoney 的取值范围从−214 748.3648 到+214 748.3647，占用 4 个字节，前面两个字节表示货币值的整数部分，后面两个字节表示小数部分。

三、字符串数据类型

SQL Server 2008 提供了 9 种字符串数据类型，可以分为 3 类，即普通字符串、Unicode 字符串和二进制字符串。

1. 普通字符串

普通字符串类型包括固定长度字符串类型 char(n)、可变长度字符串类型 varchar(n)和文本类型 text。

char(n) 是一种固定长度的非 Unicode 的字符数据，其长度为 *n* 个字节的，*n* 必须是一个介于 1 和 8 000 之间的数值，数据的存储大小为 *n* 个字节。如果没有在数据定义或变量声明语句中指定 *n*，则默认长度为 1。当一个列中包含字符串数据且每个数据项具有相同的固固定长度度时，使用 char 数据类型是一个好的选择。对于一个 char 列，不论用户输入的字符串有多长，都将固定占用 *n* 个字节的存储空间。当输入字符串的长度小于 *n* 时，如果该列不允许 NULL 值，则不足部分用空格填充；如果该列允许 NULL 值，则不足部分不再用空格填充。如果输入字符串的长度大于 *n*，则多余部分会被截断。

varchar(*n*|max) 是一种可变长度的非 Unicode 的字符数据，其长度为 *n* 个字节，*n* 必须是一个介于 1 和 8 000 之间的数值，max 指示最大存储大小是 $2^{31}-1$ 个字节。存储大小是输入数据的实际长度加 2 个字节。所输入数据的长度可以为 0 个字符。如果没有在数据定义或变量声明语句中指定 *n*，则默认长度为 1。如果一个 varchar 数据类型的列中包含有尾随空格，则这些空格会被自动删除。这是使用 varchar 数据类型的一个优点。但在使用 varchar 数据类型时，由于数据项长度可以变化，在处理速度上往往不及固定长度的 char 数据类型。

对于如何选用 char 和 varchar 有以下建议：如果列数据项的大小一致，则使用 char；如果列数据项的大小差异相当大，则使用 varchar；如果列数据项大小相差很大，而且大小可能超过 8 000 字节，可使用 varchar(max)；此外，如果要支持多种语言，可考虑使用 Unicode nchar 或 nvarchar 数据类型，以最大限度地消除字符转换问题。

text 是一种服务器代码页中长度可变的非 Unicode 数据，最大长度为 $2^{31}-1$（2 147 483 647）个字符。当服务器代码页使用双字节字符时，存储量仍是 2 147 483 647 字节。存储大小可能小于 2 147 483 647 字节（取决于字符串）。实际上，在 text 类型列中仅存储一个指针，它指向由若干个以 8KB 为单位的数据页所组成的连接表，系统经由这种连接表来存取所有的文本数据。Microsoft 建议，应尽量避免使用 text 数据类型，而应该使用 varchar(max)来存储大文本数据。

2. Unicode 字符串

Unicode 字符串类型包括固定长度字符串类型 nchar(*n*)、可变长度字符串类型 nvarchar(*n*)和文本类型 ntext。支持国际化客户端的数据库应始终使用 Unicode 数据。

nchar(*n*) 是一种长度固定的 Unicode 字符数据，其中包含 *n* 个字符的，*n* 的值必须介于 1 与 4 000 之间。数据的存储大小为 n 字节的两倍。如果没有在数据定义或变量声明语句中指定 *n*，则默认长度为 1。对一个 nchar 列指定了 *n* 的数值以后，不论用户在输入多少个字符，该列都将占用 2n 个字节的存储空间。当输入字符串长度小于 *n* 时，不论该列是否允许空值，不足部分都会用空格来填充。

nvarchar(n|max) 是一种可变长度的 Unicode 字符数据，其中包含 *n* 个字符，*n* 的值必须介于 1 与 4 000 之间。max 指示最大的存储大小为 $2^{31}-1$ 字节。数据的存储字节数是所输入字符个数的 2 倍。所输入的数据字符长度可以为零。如果没有在数据定义或变量声明语句中指定 *n*，则默认长度为 1。如果一个 nvarchar 数据类型的列中包含有尾随空格，则这些空格会被自动删除。

如果列数据项的大小可能相同，可使用 nchar。如果列数据项的大小可能差异很大，可使用 nvarchar。sysname 是系统提供的用户定义数据类型，除了不以为零外，它在功能上与

nvarchar(128)相同。sysname 用于引用数据库对象名。

ntext 是一种长度可变的 Unicode 数据，其最大长度为 $2^{30}-1$（1 073 741 823）个字符。存储字节数是所输入字符个数的 2。ntext 应该使用 nvarchar(max)来代替。

3. 二进制数据类型

二进制数据类型包括固定长度二进制数据类型 binary(*n*)、可变长度二进制数据类型 varbinary(*n*) 和大量二进制数据类型 image。

binary(*n*) 是 *n* 个字节的固定长度二进制数据。*n* 的取值范围为必须 1～8 000。不论输入数据的实际长度如何，存储空间大小均为 *n* 个字节。如果输入数据超长，则多余部分会被截掉。例如，如果将表中的一个列的数据类型指定为 binary(1)，则可以存储 0x00～0xFF 范围内的数据；如果指定为 binary(2)，则可以存储 0x0000～0xFFFF 范围内的数据。

varbinary(n|max) 是 *n* 个字节的可变长度二进制数据。*n* 的取值范围为从 1～8 000。max 指示最大的存储大小为 $2^{31}-1$ 字节。存储空间大小为实际输入数据长度+2 个字节，而不是 *n* 个字节。输入的数据长度可能为 0 字节。与 binary 数据类型不同的是，使用 varbinary 数据类型时系统会将数据尾部的 00 删除掉。

image 是一种可变长度的二进制数据类型，用于存储图片数据。一个 image 数据类型的列最多可以存储 $2^{31}-1$（2 147 483 647）个字节，约为 2GB。

若列数据项的大小一致，则使用 binary；若列数据项的大小差异相当大，则使用 varbinary；当列数据条目超出 8 000 字节时，可使用 varbinary(max)。image 应使用 varbinary(max)来代替。

四、日期和时间数据类型

在 SQL Server 2008 中，提供了多种日期和时间数据类型，包括 date、datetime、datetime2、datetimeoffset、smalldatetime 以及 time。对于新的工作，建议使用 time、date、datetime2 和 datetimeoffset 数据类型，因为这些类型符合 SQL 标准，而且更易于移植。time、datetime2 和 datetimeoffset 提供更高精度的秒数，datetimeoffset 可为全局部署的应用程序提供时区支持。

1. date 数据类型

date 数据类型用于存储日期数据，存储范围为公元元年 1 月 1 日到公元 9999 年 12 月 31 日，存储空间大小为 3 个字节。

对于 date 数据类型，用于客户端的默认字符串文字格式为 YYYY-MM-DD，其中 YYYY 是表示年份的 4 位数字，范围为从 0001 到 9999；MM 是表示指定年份中的月份的两位数字，范围为从 01 到 12；DD 是表示指定月份中的某一天的两位数字，范围为从 01 到 31（最高值取决于具体月份）。默认值为 1900-01-01。

2. datetime 数据类型

datetime 数据类型用于定义一个与采用 24 小时制并带有秒小数部分的一日内时间相组合的日期，存储的日期范围为 1753 年 1 月 1 日到 9999 年 12 月 31 日，时间范围为 00:00:00 到 23:59:59.997，存储空间大小为 8 个字节。

对于 datetime 数据类型，系统的存储格式为 YYYY-MM-DD hh:mm:ss[.n*]，其中 YYYY 是表示年份的 4 位数字，范围为 1753 到 9999；MM 是表示指定年份中的月份的两位数字，范围为 01 到 12；DD 是表示指定月份中的某一天的两位数字，范围为 01 到 31（最高值取决于相应月份）；其中 hh 是表示小时的两位数字，范围为 00 到 23；mm 是表示分钟的两位数字，范围为 00 到 59；ss 是表示秒钟的两位数字，范围为 00 到 59；n*为一个 0 到 3 位的数字，范围

为 0 到 999，表示秒的小数部分。datetime 数据类型的默认值为 1900-01-01 00:00:00。可以使用 SET DATEFORMAT 语句更改日期顺序。

3. datetime2 数据类型

datetime2 数据类型定义结合了 24 小时制时间的日期，可将其视作 datetime 类型的扩展，但其数据范围更大，默认的小数精度更高，并具有可选的用户定义的精度。

datetime2 存储的日期范围为 0001-01-01 到 9999-12-31，即公元元年 1 月 1 日到公元 9999 年 12 月 31 日，时间范围为 00:00:00 到 23:59:59.9999999，用于客户端默认的字符串文字格式为 YYYY-MM-DD hh:mm:ss[.n*], 其中 YYYY 是一个 4 位数, 范围从 0001 到 9999, 表示年份；MM 是一个两位数，范围从 01 到 12，表示指定年份中的月份；DD 是一个两位数，范围为 01 到 31（具体取决于月份），表示指定月份中的某一天；hh 是一个两位数，范围从 00 到 23，表示小时；mm 是一个两位数，范围从 00 到 59，表示分钟；ss 是一个两位数，范围从 00 到 59，表示秒钟；n*代表 0 到 7 位数字，范围从 0 到 9999999，表示秒的小数部分。当精度小于 3 时存储空间大小为 6 个字节；当精度为 4 和 5 时为存储空间大小为 7 个字节；所有其他精度则需要 8 个字节。

4. datetimeoffset 数据类型

datetimeoffset 数据类型用于定义一个与采用 24 小时制并可识别时区的一日内时间相组合的日期, 存储的日期范围为公元元年 1 月 1 日到公元 9999 年 12 月 31 日, 时间范围为 00:00:00 到 23:59:59.9999999，时区偏移量范围为-14:00 到+14:00。存储空间大小为 10 个字节，默认值为 1900-01-01 00:00:00 00:00。

对于 datetimeoffset 数据类型而言，用于客户端的默认字符串文字格式为 YYYY-MM-DD hh:mm:ss[.nnnnnnn] [{+|−}hh:mm], 其中 YYYY 是表示年份的 4 位数字, 范围为 0001 到 9999；MM 是表示指定年份中的月份的两位数字，范围为 01 到 12；DD 是表示指定月份中的某一天的两位数字，范围为 01 到 31（最高值取决于相应月份）；hh 是表示小时的两位数字，范围为 00 到 23；mm 是表示分钟的两位数字，范围为 00 到 59；ss 是表示秒钟的两位数字，范围为 00 到 59；n*是 0 到 7 位数字，范围为 0 到 9999999，表示秒的小数部分；第二个方括号内给出相对于 UTC 时间的时区偏移量，其中 hh 是两位数，范围为−14 到+14，表示时区偏移量中的小时数；mm 是两位数，范围为 00 到 59，表示时区偏移量中的额外分钟数。时区偏移量中必须包含+（加）或−（减）号，这两个符号表示是在 UTC 时间的基础上加上还是从中减去时区偏移量以得出本地时间。时区偏移量的有效范围为−14:00 到+14:00。

5. smalldatetime 数据类型

smalldatetime 数据类型定义结合了一天中的时间的日期，此时间为 24 小时制，秒始终为零（:00），并且不带秒小数部分。

smalldatetime 数据类型存储的日期范围为 1900 年 1 月 1 日到 2079 年 6 月 6 日，时间范围为 00:00:00 到 23:59:59。系统的存储格式为 YYYY-MM-DD hh:mm:ss，其中 YYYY 是表示年份的 4 位数字，范围为 1900 到 2079，MM 是表示指定年份中的月份的两位数字，范围为 01 到 12 ;DD 是表示指定月份中的某一天的两位数字, 范围为 01 到 31（最高值取决于相应月份）；hh 是表示小时的两位数字，范围为 00 到 23；mm 是表示分钟的两位数字，范围为 00 到 59；ss 是表示秒钟的两位数字，范围为 00 到 59。存储空间的大小为 4 个字节。

6. time 数据类型

time 数据类型定义一天中的某个时间，此时间不能感知时区且基于 24 小时制。存储的时

间范围为 00:00:00.0000000 到 23:59:59.9999999。

对于 time 数据类型，用于客户端默认的字符串文字格式为 hh:mm:ss[.nnnnnnn]，其中 hh 是表示小时的两位数字，范围为 0 到 23；mm 是表示分钟的两位数字，范围为 0 到 59；ss 是表示秒的两位数字，范围为 0 到 59；n*是 0 到 7 位数字，范围为 0 到 9999999，表示秒的小数部分。

五、其他数据类型

在 SQL Server 2008 中，还提供了几种比较特殊的数据类型，包括 cursor、hierarchyid、sql_variant、table、timestamp、uniqueidetifier 以及 xml。使用这些数据类型可以完成特殊数据对象的定义和存储。

1. cursor 数据类型

这是变量或存储过程 OUTPUT 参数的一种数据类型，这些参数包含对游标的引用。使用 cursor 数据类型创建的变量可以为空。在 Transact-SQL 语句中，有些操作可以引用那些带有 cursor 数据类型的变量和参数。但要特别注意，对于 CREATE TABLE 语句中的列，是不能使用 cursor 数据类型的。

2. hierarchyid 数据类型

hierarchyid 数据类型是一种长度可变的系统数据类型。使用 hierarchyid 可以表示层次结构中的位置。类型为 hierarchyid 的列不会自动表示树。由应用程序来生成和分配 hierarchyid 值，使行与行之间的所需关系反映在这些值中。hierarchyid 的值具有以下属性：非常紧凑；按深度优先顺序进行比较；支持任意插入和删除。

3. sql_variant 数据类型

这种数据类型用于存储 SQL Server 支持的各种数据类型的值，但 text、ntext、image、timestamp 和 sql_variant 除外。sql_variant 数据类型可以用在列、参数和变量中并返回用户定义函数的值。sql_variant 允许这些数据库对象支持其他数据类型的值。sql_variant 数据类型的最大长度可达 8 016 字节。

4. table 数据类型

这是一种特殊的数据类型，用于存储结果集以供后续处理。该数据类型主要用于临时存储一组记录，这些记录将作为表值函数的结果集返回。

5. timestamp 数据类型

这是公开数据库中自动生成的唯一二进制数字的数据类型。timestamp 通常用作给表行加版本戳的机制，存储大小为 8 个字节。当把表中的一个列指定为 timestamp 数据类型时，如果设置该列不允许空值，则它等价于定长二进制数据类型 binary(8)；如果允许空值，则等价于定长二进制数据类型 varbinary(8)。timestamp 列的数值格式类似于 0x000000000000025A。一个表中只能有一个 timestamp 类型的列。一旦将表中的一个列指定为 timestamp 数据类型，该列的值就不用设置了，它会随着表中记录内容的修改而自动更新，而且在整个数据库范围都是唯一的。

使用某一行中的 timestamp 列可以很容易地确定该行中的任何值自上次读取以后是否发生了更改。如果对行进行了更改，就会更新该时间戳值。如果没有对行进行更改，则该时间戳值将与以前读取该行时的时间戳值一致。如果要获取数据库的当前时间戳值，可以使用@@DBTS 函数。

6. uniqueidetifier 数据类型

uniqueidentifier 即数据类型全局性唯一标识数据类型，用于存储一个由 16 个字节组成的二进制数字，其数值格式类似于 BA9B4EF2-8F8B-4B42-8D0A-E0994AABD7FC，该识别码称为全局性唯一标识符（GUID，Globally Unique Identifier）。

若要对一个 uniqueidentifier 列或局部变量进行初始化，通常使用以下两种方法。

- 使用 NEWID 函数产生 GUID。
- 将一个“ xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx ”形式的字符串常量转换为 GUDI，其中 x 是一个 16 进制数字，取值范围为 0～9、a～f。

具有更新订阅的合并复制和事务复制使用 uniqueidentifier 列来确保在表的多个副本中唯一地标识行。

7. xml 数据类型

这是一种存储 XML 数据的数据类型。使用 xml 数据类型可以在 SQL Server 数据库中存储 XML 文档和片段。XML 片段是缺少单个顶级元素的 XML 实例。可以创建 xml 类型的列和变量，并在其中存储 XML 实例。xml 数据类型实例的存储表示形式不能超过 2GB。

六、别名数据类型

别名数据类型类型是基于 SQL Server 中的系统数据类型创建的一种用户定义数据类型（UDDT）。当多个表必须在一个列中存储相同类型的数据，而且必须确保这些列具有相同的数据类型、长度和为空性时，可以使用别名类型。例如，如果在学生信息管理数据库的几个表中都包含班级编号列，则可以基于 char 数据类型创建一个名为 class_number 的别名类型。自 SQL Server 2005 起，表变量中开始支持没有规则和附加的默认定义的别名类型。在 SQL Server 2005 之前，SQL Server 不支持表变量中的别名类型。

创建别名数据类型时，必须提供下列参数：名称、新数据类型基于的系统数据类型以及为空性（数据类型是否允许空值）。如果未明确定义为空性，系统将基于数据库或连接的 ANSI NULL 默认设置进行指定。

如果别名类型是在 model 系统数据库中创建的，则它将存在于所有用户定义的新数据库中。但是，如果数据类型是在用户定义的数据库中创建的，该数据类型将只存在于该用户定义的数据库中。

在 SQL Server 2008 中，可以使用 SQL Server Management Studio 工具来创建和删除别名数据类型，也可以使用 CREATE TYPE 和 DROP TYPE 语句来创建和删除别名数据类型。对于已存在的别名数据类型，可在创建和修改表的过程使用。

CREATE TYPE 语句用于创建别名数据类型，语法格式如下：

```
CREATE TYPE [schema_name.]type_name
{
   FROM base_type
   [(precision[,scale])]
   [NULL|NOT NULL]
}[;]
```

其中参数 *schema_name* 为别名数据类型或用户定义类型所属架构的名称。*type_name* 指定别名数据类型或用户定义类型的名称，该名称必须符合标识符规则。*base_type* 指定别名数据类

型所基于的数据类型，由 SQL Server 提供。base_type 也可以是映射到这些系统数据类型之一的任何数据类型同义词。*precision* 和 *scale* 适用于 *base_type* 为 decimal 或 numeric 时，它们的值为非负整数，*precision* 指示可保留的十进制数字位数的最大值，包括小数点左边和右边的数字；*scale* 指示十进制数字的小数点右边最多可保留多少位，它必须小于或等于精度值。

NULL | NOT NULL 指定此类型是否可容纳空值。如果未指定，则默认值为 NULL。

若要在对象资源管理器中删除别名数据类型，可以在目录树中展开“用户定义数据类型”结点，右键单击该别名数据类型并选择“删除”。

也可以使用 DROP TYPE 语句从当前数据库中删除别名数据类型，语法格式如下：

```
DROP TYPE [schema_name.]type_name[;]
```

其中参数 *schema_name* 为别名数据类型所属的架构名。*type_name* 指定要删除的别名数据类型或用户定义的类型的名称。

任务 3　创建表

任务描述

在本任务中，首先创建一个名为“学生成绩”的数据库，然后使用 CREATE TABLE 语句在这个数据库中创建以下两个表（如图 3.4 所示）。

(1)“系别”表，包含两个列：“系别编号”列为 dep_num 类型，设为标识列，种子和增量均为 1；“系别名称”列为 varchar 类型，长度为 50，不允许为空。

(2)“班级”表，包含 3 个列：“班级编号”列为 class_num 类型；“系别编号”列为 dep_num 类型；“专业名称”列为 varchar 类型，长度为 50，不允许为空。

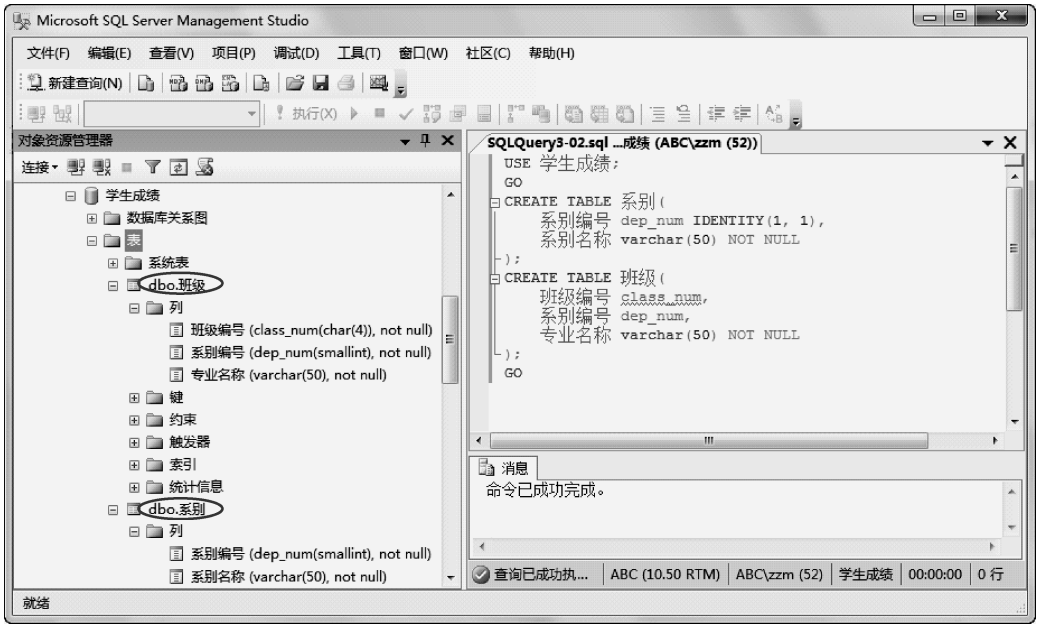

图 3.4　“班级”表和“系别”表

任务实现

实现步骤如下：

（1）在对象资源管理器中，连接到数据库引擎实例。

（2）新建一个查询，然后在查询编辑器窗口中编写以下 SQL 语句：

```
USE 学生成绩;
GO
CREATE TABLE 系别(
    系别编号 dep_num IDENTITY(1, 1),
    系别名称 varchar(50) NOT NULL
);
CREATE TABLE 班级(
    班级编号 class_num,
    系别编号 dep_num,
    专业名称 varchar(50) NOT NULL
);
GO
```

（3）将脚本文件保存为 SQLQuery3-02.sql，按 F5 键执行脚本。

（4）在对象资源管理器中，单击“数据库”，然后从“查看”菜单中选择“刷新”命令，并查看新建的数据库、用户表及其各个列。

相关知识

数据通常存储于永久表中，不过也可以根据需要来创建临时表。表存储于数据库文件中，任何拥有权限的用户均可对其进行操作。创建表之后，根据需要还可以对最初创建表时定义的许多选项进行修改。

在 SQL Server 2008 中，创建表主要有以下两种方法：一种方法使用表设计器图形工具创建表，另一种方法使用 CREATE TABLE 语句创建表。

一、使用表设计器创建表

使用表设计器可以创建新表并对该表进行命名，然后将其添加到现有数据库中。具体实现步骤如下：

（1）在对象资源管理器中，连接到数据库引擎，然后展开该实例。

（2）展开要在其中创建表的数据库，右键单击“表”，然后从快捷菜单中选择“新建表”命令，如图 3.5 所示。

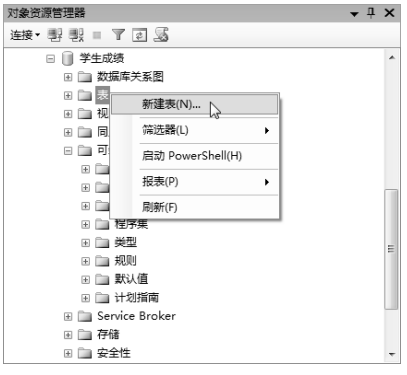

图 3.5　在数据库中创建表

（3）在表设计器上部的网格中输入列名称，从“数据类型”列表中为列选择一种数据类型，在“允许 Null 值”列中选择或清除复选框，指定列是否允许空值，如图 3.6 所示。

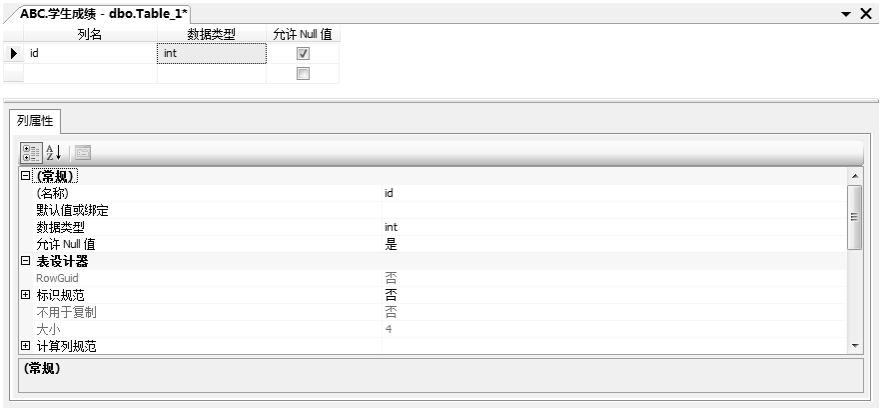

图 3.6　表设计器

（4）根据需要，在表设计下部的“ 列属性 ”列表中设置列的附加属性，例如默认值、精度、小数位数、是否标识列、标识增量和标识种子等。

（5）重复步骤（3）和（4），在表中定义更多的列。

（6）从“ 文件 ”菜单中选择“ 保存 ”命令，或在工具栏上单击“ 保存 ”按钮。

（7）在如图 3.7 所示的“ 选择名称 ”对话框中为输入表名称，然后单击“ 确定 ”按钮。

图 3.7　指定表名称

二、使用 CREATE TABLE 语句创建表

CREATE TABLE 语句用于在当前数据库或指定数据库中创建新表，基本语法格式如下：

```
CREATE TABLE
    [database_name.[schema_name].|schema_name.]table_name
        ({<column_definition>}[,...n])[;]

<column_definition>::=
column_name data_type
    [NULL|NOT NULL]
    [
        [CONSTRAINT constraint_name]DEFAULT constant_expression]
        |[IDENTITY[(seed,increment)]
    ]
    [ROWGUIDCOL][<column_constraint>[...n]]
```

其中参数 *database_name* 指定在其中创建表的数据库的名称。*database_name* 必须指定现有数据库的名称。如果未指定，则 *database_name* 默认为当前数据库。

schema_name 指定新表所属架构的名称。架构是指包含表、视图、过程等的容器。它位于数据库内部，而数据库位于服务器内部。特定架构中的每个安全对象都必须有唯一的名称。架构中安全对象的完全指定名称包括此安全对象所在的架构的名称。在 SQL Server 2005 和 SQL Server 2008 中，架构既是一个容器，又是一个命名空间。

table_name 指定新表的名称。表名必须遵循标识符规则。除了本地临时表名（以单个数字

符号#为前缀的名称）不能超过 116 个字符外，*table_name* 最多可包含 128 个字符。对于每一个架构在一个数据库内表的名称必须是唯一的，但如果为每张表指定了不同的架构，则可以创建多个具有相同名称的表。

column_name 指定表中列的名称。列名必须遵循标识符规则，并在表中唯一。*column_name* 可包含 1～128 个字符。对于使用 timestamp 数据类型创建的列，可省略 *column_name*。若未指定 *column_name*，则 timestamp 列的名称将默认为 timestamp。每个表至多可定义 1 024 列。

type_name 指定列的数据类型，数据类型可以是系统数据类型，也可以是基于系统数据类型的别名类型。

NULL | NOT NULL 确定列中是否允许使用空值。

CONSTRAINT 为可选关键字，表示 PRIMARY KEY、NOT NULL、UNIQUE、FOREIGN KEY 或 CHECK 约束定义的开始。*constraint_name* 表示约束的名称。约束名称必须在表所属的架构中唯一。

DEFAULT 指定列的默认值。若在插入记录的过程中未显式提供值，则该列将获得此默认值。DEFAULT 定义可适用于除定义为 timestamp 或带 IDENTITY 属性的列以外的任何列。*constant_expression* 是用作列的默认值的常量、NULL 或系统函数。

IDENTITY 表示新列是标识列。在表中添加新行时，数据库引擎将为该列提供一个唯一一的增量值。标识列通常与 PRIMARY KEY 约束一起用作表的唯一行标识符。可将 IDENTITY 属性分配给 tinyint、smallint、int、bigint、decimal(p,0)或 numeric(p,0)列。对于每个表，只能创建一个标识列。不能对标识列使用绑定默认值和 DEFAULT 约束。*seed* 是装入表的第一行所使用的值。*increment* 是向装载的前一行的标识值中添加的增量值。必须同时指定种子和增量，或者两者都不指定。如果二者均未指定，则取默认值(1,1)。

ROWGUIDCOL 指示新列是行 GUID 列。对于每个表，只能将其中的一个 uniqueidentifier 列指定为 ROWGUIDCOL 列。ROWGUIDCOL 属性只能分配给表中的 uniqueidentifier 列。ROWGUIDCOL 属性并不强制列中所存储值的唯一性。该属性也不会为插入到表中的新行自动生成值。<column_constraint>表示列约束。

任务 4　在表中添加列

任务描述

在本任务中，要求在“学生成绩”数据库中执行以下操作。

（1）创建一个名称为“课程”的表，在该表中定义以下两个列：“课程编号”列的数据类型为 int；“课程名称”为 varchar 类型，长度为 20，这两个列都不允许为空。

（2）创建“课程”表后向该表中添加以下两个列：“课程类别”列为 char 类型，长度为 8；“考试类别”列为 char 类型，长度为 4，这两个列都不允许为空。

任务分析

使用 CREATE TABLE 语句创建表后，可以使用 ALTER TABLE 语句向该表中添加列。要求所添加的列必须允许空值或对列创建 DEFAULT 约束。由于新创建的“课程”表目前是一个不包

含任何数据行的空表，因此可以对新添加的列设置 NOT NULL。

任务实现

实现步骤如下：

（1）在对象资源管理器中，连接到数据库引擎实例。

（2）新建一个查询，然后在查询编辑器窗口中编写以下语句：

```
USE 学生成绩;
GO
CREATE TABLE 课程(
    课程编号 int NOT NULL,
    课程名称 varchar(20) NOT NULL
);
GO
ALTER TABLE 课程
ADD 课程类别 char(8) NOT NULL,
    考试类别 char(4) NOT NULL;
GO
```

（3）将脚本文件保存为 SQLQuery3-03.sql，按 F5 键执行脚本。

（4）在对象资源管理器中展开“ 课程 ”表的“ 列 ”结点，以查看列定义；或者在表设计器中查看该表的结构。

相关知识

在 SQL Server 中，如果列允许空值或对列创建 DEFAULT 约束，则可以将列添加到现有表中。将新列添加到表时，数据库引擎在该列为表中的每个现有数据行插入一个值。因此，在向表中添加列时对列添加 DEFAULT 定义会很有用。如果新列没有 DEFAULT 定义，则必须指定该列允许空值。数据库引擎将空值插入该列，如果新列不允许空值，则返回错误。也可以从现有表中删除列，但具有下列特征的列除外：用于索引；用于 CHECK、FOREIGN KEY、UNIQUE 或 PRIMARY KEY 约束；与 DEFAULT 定义关联或绑定到某一默认对象；绑定到规则；已注册支持全文；用作表的全文键。

一、使用表设计器添加或删除表列

若要在表中添加或删除列，可以使用表设计器来实现，操作步骤如下。

（1）在对象资源管理器中，右键单击要向其添加列的表，然后选择“ 设计 ”命令，如图 3.8 所示。

（2）在表设计器中打开该表后，若要向表中添加列，可执行下列操作之一：若要在表的末尾添加列，可将光标置于“ 列名 ”列的第一个空白单元格中；若要在某列前面插入一列，可用右键单击网格中表示该列的行，然后选择“ 插入列 ”，如图 3.9 所示。

（3）在“ 列名 ”列的单元格中输入列名。

（4）按 Tab 键转到“ 数据类型 ”单元格，并从下拉列表中选择所需的数据类型。