

第3章 操作数据表

创建数据库以后，接下来的工作就是创建和管理数据表。数据表是数据库的基本构成单元，是数据库中最重要对象，用来保存用户的各类数据，后期的各种操作也是在数据表的基础上进行的，因此对表的管理是对 SQL Server 数据库管理的重要内容。在实际应用中，数据表常用的操作有创建、删除、数据管理以及约束等，本节将重点介绍如何在 SQL Server 2012 中操作数据库表。

本章学习要点：

- ☐ 了解 SQL Server 中的系统表
- ☐ 掌握创建数据表的几种方式
- ☐ 熟悉表和列的相关操作
- ☐ 掌握列的数据类型
- ☐ 掌握主键约束和自动增长标识
- ☐ 掌握唯一性约束
- ☐ 了解空与非空约束
- ☐ 熟悉默认值约束和检查约束
- ☐ 掌握外键约束

3.1 了解表

表是 SQL Server 数据库中最主要的数据库对象，用于存储数据库中的所有数据。在介绍用户如何创建表之前，首先了解一下数据表的概念和常用的一些系统表。

3.1.1 表的概念

数据库中的所有数据存储在表中，数据表包括行和列。列决定表中数据的类型，行包含实际的数据。数据在表中的组织方式与在电子表格中相似，都是按行和列的格式组织的。每一行代表一条唯一的记录，每一列代表记录中的一个字段。例如，图 3-1 是一个基本的员工信息表，表中的每一行代表一条人员信息，每一列代表人员的详细资料，如编号、名称、年龄、联系电话、出生日期、进公司工作的日期以及个人说明等。

数据表代表实体，并且有唯一的名称，由该名称确定实体。例如，employee 表表示员工实体，在该实例中保存每名员工的基本信息。

数据表由行和列组成，行有时也称为记录，列有时也称为字段或者域。表中每一列都有一个列名用来描述该列的特性。每个表包含若干行，表的第一行为各列标题，其余

行都是数据。

	employNo	employName	employAge	employPhone	employBirth	employEnter	employIntro
	1001	admin	59	13838966696	1955-10-10 0...	1975-10-01 0...	
	1002	Locy	24	15890023651	1990-01-21 0...	2012-10-01 0...	
	1003	张谦	28	15890026358	1986-03-21 0...	2010-12-12 0...	
	1004	陈海风	32	15890036258	1982-01-07 0...	2001-12-12 0...	副经理
	1005	李燕	28	18703625985	1986-05-26 0...	2010-12-12 0...	
▶*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

图 3-1 employee 表

在表中，行和列的顺序可以任意。但是对于每一个表，最多可以定义 1024 列。在同一个表中列名必须唯一。在定义表时，用户还必须为每一个列指定一种数据类型。



注意

列名在表中的唯一性是由 SQL Server 2012 强制实现的。而行在表中的唯一性是由用户通过增加主键约束来强制实现的，即在一个表中，为了满足实际应用的需要，两行相同的记录毫无意义。

3.1.2 系统表

SQL Server 2012 通过一系列表来存储所有对象、数据类型、约束、配置选项、可利用资源的相关信息，这一系列表被称为系统表。任何用户都不应该直接更改系统表，例如，不要尝试使用 DELETE、UPDATE 和 INSERT 语句或用户定义的触发器修改系统表。将系统表分类时，可以将其分为备份和还原表、日志传送表、变更数据捕获表、复制表、数据库维护计划表、SQL Server 代理表以及 SQL Server 扩展事件表等。

系统表中还有一种系统基表，它是基础表，用于实际存储特定数据库的元数据。在该方面，master 数据库有些特别，因为它包含一些在其他任何数据库中都找不到的表，这些表包含服务器范围内的持久化元数据。例如，表 3-1 中列出了 SQL Server 2012 中的部分系统基表，并对它们进行说明。

表 3-1 SQL Server 2012 中的部分系统基表

系统基表	说明
sys.syssschobjs	存在于每个数据库中。每一行表示数据库中的一个对象
sys.sysbinobjs	存在于每个数据库中。数据库中的每个 Service Broker 实体都存在对应的一行。Service Broker 实体包括消息类型、服务合同和服务
sys.sysclsobjs	存在于每个数据库中。共享相同通用属性的每个分类实体均存在对应的一行，这些属性包括程序集、备份设备、全文目录、分区函数、分区方案、文件组和模糊处理键
sys.sysnsobjs	存在于每个数据库中。每个命名空间范围内的实体均存在对应的一行。此表用于存储 XML 集合实体

续表

系统基表	说明
sys.sysiscols	存在于每个数据库中。每个持久化索引和统计信息列均存在对应的一行
sys.sysscalartypes	存在于每个数据库中。每个用户定义类型或系统类型均存在对应的一行
sys.sysdbreg	仅存在于 master 数据库中。每个注册数据库均存在对应的一行
sys.sysxsrvs	仅存在于 master 数据库中。每个本地服务器、链接服务器或远程服务器均存在对应的一行
sys.sysxlgns	仅存在于 master 数据库中。每个服务器主体均存在对应的一行
sys.sysusermsg	仅存在于 master 数据库中。每一行表示用户定义的错误消息
sys.ftinds	存在于每个数据库中。数据库中的每个全文索引均存在对应的一行
sys.sysxprops	存在于每个数据库中。每个扩展属性均存在对应的一行
sys.sysallocunits	存在于每个数据库中。每个存储分配单元均存在对应的一行
sys.sysrowsets	存在于每个数据库中。索引或堆的每个分区行集均存在对应的一行
sys.sysrowsetrefs	存在于每个数据库中。行集引用的每个索引均存在对应的一行
sys.sysobjvalues	存在于每个数据库中。实体的每个常规值属性均存在对应的一行
sys.sysguidrefs	存在于每个数据库中。每个 GUID 分类 ID 引用均存在对应的一行



提示

SQL Server 2012 中包含多种系统表和多个系统基表，表 3-1 中只列出了部分系统基表，关于其他的系统基表和系统表，读者可以在 SQL Server 2012 在线文档上查找到更多资料。

3.2 创建表

在 SQL Server 2012 中创建表的常用两种方式是通过图形界面创建表和通过 Transact-SQL 语句创建表，下面分别介绍这两种方式。

3.2.1 图形界面创建

在一个基本的学生管理系统中，需要包含学生的基本信息，如学生编号、姓名、性别、年龄、出生日期、联系电话以及家庭住址等。在范例 1 中介绍如何通过图形界面工具创建数据表。

【范例 1】

通过图形界面在 studenttest 数据库中创建数据表的一般步骤如下。

- (1) 展开【对象资源管理器】窗格中 studenttest 数据库下的【表】节点。因为 studenttest 数据库尚未添加新表，因此展开它的子选项时只存在【系统表】节点，如图 3-2 所示。
- (2) 右键单击【表】选项，在弹出的快捷菜单中选择【新建表】命令，这时会在对象资源管理器右侧出现编辑窗口，如图 3-3 所示。
- (3) 可以在编辑窗口中输入或选择表的列名、数据类型、是否允许为 Null 值。例如，在如图 3-3 所示的编辑窗口中添加 stuNo 列，指定该列的数据类型为 nvarchar(10)，且不允许为 Null 值，如图 3-4 所示。

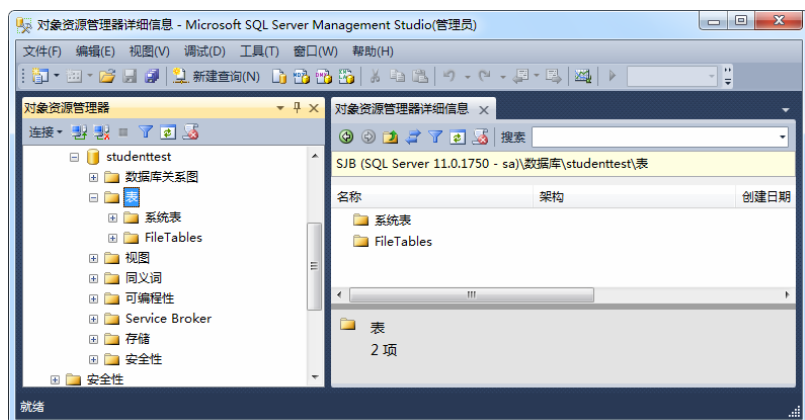


图 3-2 创建 studenttest 数据库

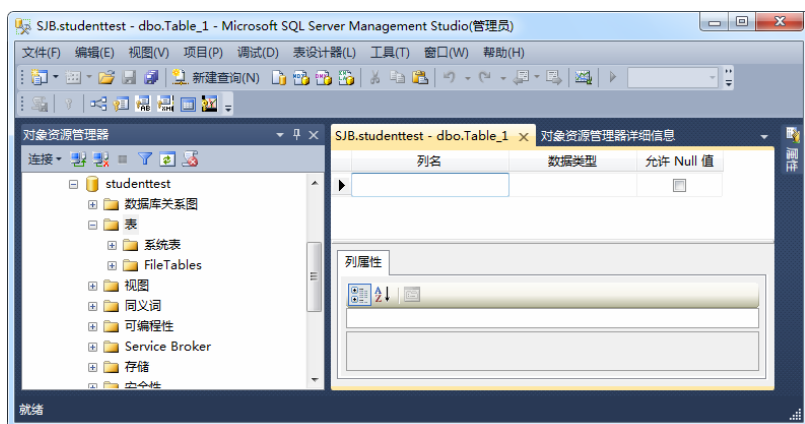


图 3-3 新建表时的编辑窗口

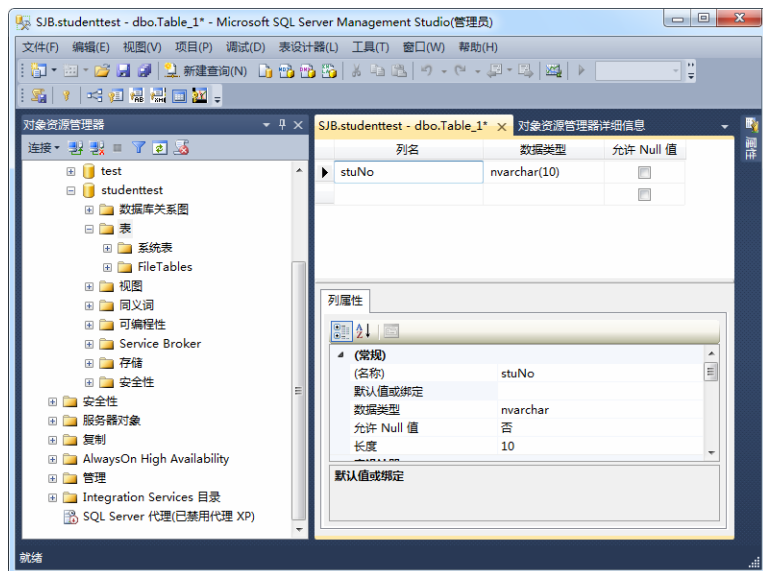


图 3-4 为表添加 stuNo 列

从图 3-4 中可以看出,在表中添加字段列后,在编辑窗口的底部设置列的属性,如名称、默认值、数据类型和长度等。

(4) 在表中继续添加剩余的列,包括 stuName、stuAge、stuAge、stuBirth、stuPhone 以及 stuAddress,并完成相关数据类型的设置。

(5) 完成所有列的添加后,需要考虑对列的主键的设置。选定选作主键的列(如 stuNo),右键单击列名弹出如图 3-5 所示的快捷菜单。

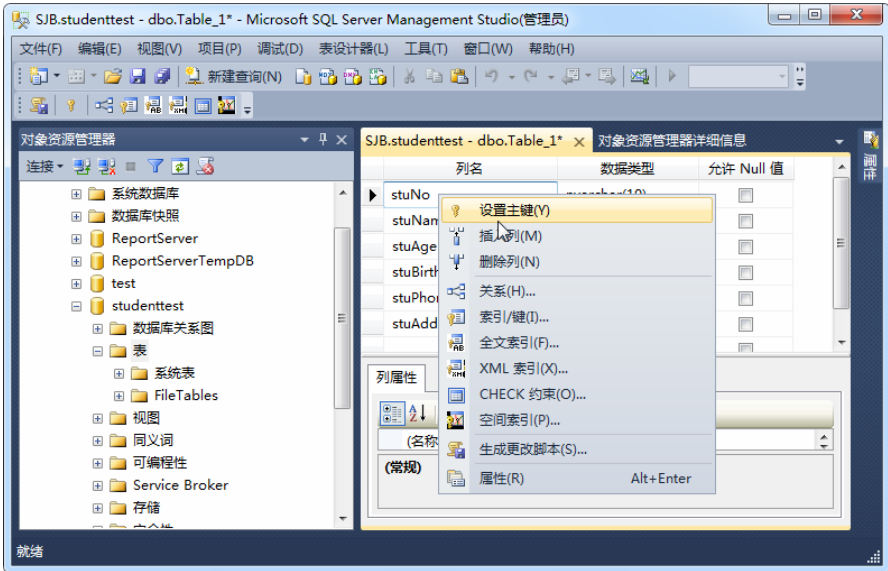


图 3-5 将字段列设置为主键

(6) 在图 3-5 中选择【设置主键】命令完成主键的设置,这时会在 stuNo 列的前面出现一个钥匙标记,这说明该列已经被设置为主键,如图 3-6 所示。

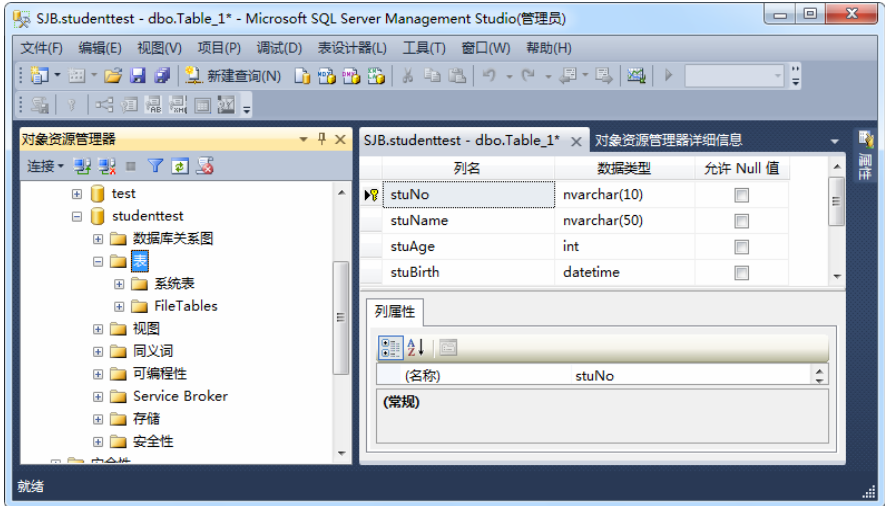


图 3-6 完成对主键列的添加

(7) 单击工具栏中的【保存】按钮或者按 Ctrl+S 键, 弹出【选择名称】对话框, 如图 3-7 所示。在对话框中输入表的名称 StudentBaseInfo, 然后单击【确定】按钮保存表。

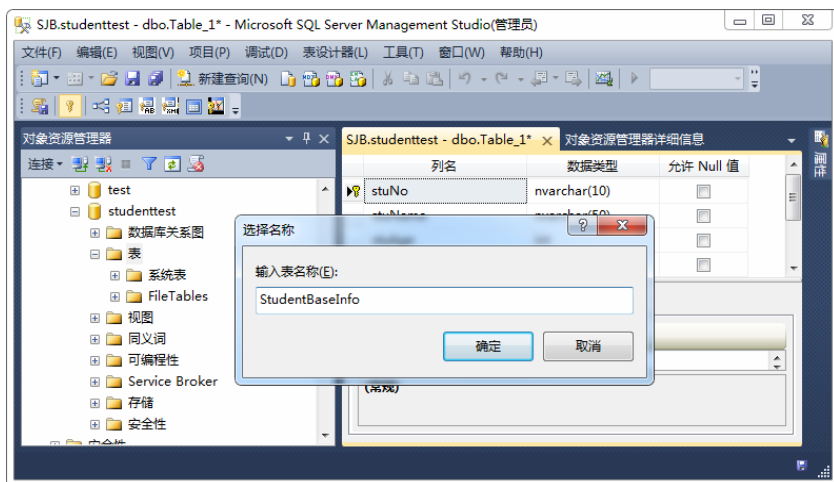


图 3-7 保存表时输入表的名称

(8) 完成保存后, 在【对象资源管理器】窗格中将看到创建的 StudentBaseInfo 表, 单击该表可以查看表中的各个节点 (如列、键、约束和索引等), 如图 3-8 所示。

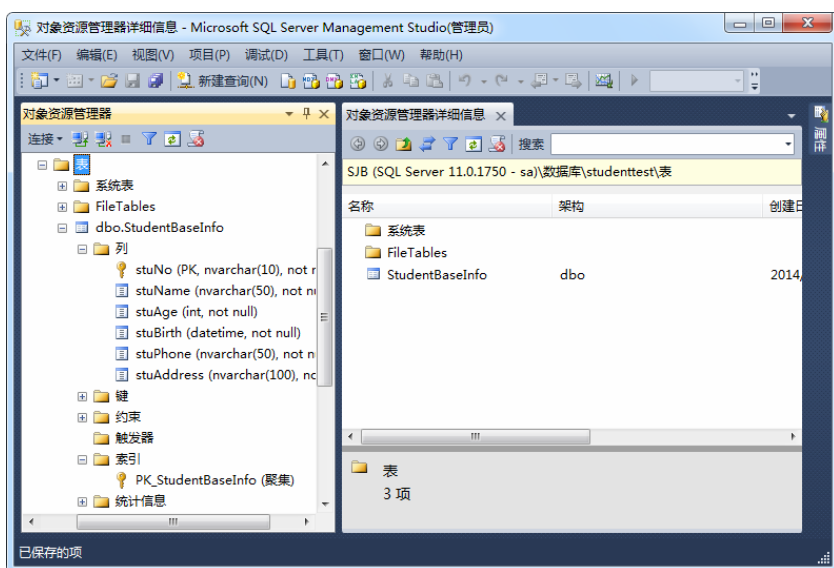


图 3-8 查看表中的信息

3.2.2 Transact-SQL 语句创建

除了使用图形界面创建数据表外, 还经常使用 Transact-SQL 语句进行创建。通过 Transact-SQL 语句创建表时需要使用 CREATE TABLE 命令。完整语法如下:

```

CREATE TABLE
    [ database_name . [ schema_name ] . | schema_name . ] table_name
    [ AS FileTable ]
    ( { <column_definition> | <computed_column_definition>
        | <column_set_definition> | [ <table_constraint> ] [ ,...n ] } )
    [ ON { partition_scheme_name ( partition_column_name ) | filegroup |
        "default" } ]
    [ { TEXTIMAGE_ON { filegroup | "default" } } ]
    [ FILESTREAM_ON { partition_scheme_name | filegroup | "default" } ]
    [ WITH ( <table_option> [ ,...n ] ) ]
[ ; ]

```

上述语法的说明如下。

(1) **database_name**: 表示在其中创建表的数据库的名称, 它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。

(2) **schema_name**: 创建数据库表的所有者名, 如果为空, 则默认新表的创建者在当前数据库中的用户名。

(3) **table_name**: 创建数据表的名称, 表名必须遵循有关标识符的规则。

(4) **AS FileTable**: 将新表创建为 FileTable, 用户无须指定列, 因为 FileTable 具有固定架构。

(5) **<column_definition>**: 表示数据列的语句结构。

(6) **<table_constraint>**: 表示对数据表的约束进行设置。

<column_definition>用于定义数据列的语句结构。它的完整语法如下:

```

<column_definition> ::=
column_name <data_type>
    [ FILESTREAM ]
    [ COLLATE collation_name ]
    [ SPARSE ]
    [ NULL | NOT NULL ]
    [
        [ CONSTRAINT constraint_name ] DEFAULT constant_expression ]
    | [ IDENTITY [ ( seed ,increment ) ] [ NOT FOR REPLICATION ]
    ]
    [ ROWGUIDCOL ]
    [ <column_constraint> [ ...n ] ]
<data type> ::=
[ type_schema_name . ] type_name
    [ ( precision [ , scale ] | max |
        [ { CONTENT | DOCUMENT } ] xml_schema_collection ) ]

```

上述语法的说明如下。

(1) **column_name**: 表中列的名称。

(2) **<data type>**: 定义列的数据类型。在它的语法中, [type_schema_name.]type_name 指定列的数据类型以及该列所属的架构; **precision** 和 **scale** 分别表示指定数据类型的精度

和指定数据类型的小数位数。

(3) **COLLATE collation_name**: 指定列的排序规则。

(4) **NULL|NOT NULL**: 确定列中是否允许有空值。

(5) **CONSTRAINT**: 可选关键字, 表示 **PRIMARY KEY**、**NOT NULL**、**UNIQUE**、**FOREIGN KEY** 或 **CHECK** 约束定义的开始。

(6) **constraint_name**: 约束的名称, 它必须在表所属的架构中唯一。

(7) **DEFAULT**: 如果在插入过程中未显式提供值, 则指定为列提供的值。

(8) **constant_expression**: 它是用作列的默认值的常量、**NULL** 或系统函数。

(9) **IDENTITY**: 指示新列是标识列, 每个表只能创建一个标识列。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。

(10) **seed**: 加载到表中的第一个行时所使用的值。

(11) **increment**: 加载到前一行的标识值中要添加的增量值。

(12) **NOT FOR REPLICATION**: 在 **CREATE TABLE** 语句中, 可为 **IDENTITY** 属性、**FOREIGN KEY** 约束和 **CHECK** 约束指定 **NOT FOR REPLICATION** 子句。如果为 **IDENTITY** 属性指定了该子句, 则复制代理执行插入时, 标识列中的值将不会增加。如果为约束指定了此子句, 则当复制代理执行插入、更新或删除操作时, 将不会强制执行此约束。

(13) **ROWGUIDCOL**: 指示新列是行 GUID (Globally Unique Identifier, 全局唯一标识符) 列。



注意

GUID 是唯一的二进制数, 世界上的任何两台计算机都不会生成重复的 GUID 值。GUID 主要用在拥有多个节点、多台计算机的网络中, 分配必须具有唯一性的标识符。

【范例 2】

了解 **CREATE TABLE** 命令的语法后, 通过 **CREATE TABLE** 语句在 **studenttest** 数据库中创建 **PurchaseOrderDetail** 表。代码如下:

```
USE studenttest;
GO
CREATE TABLE dbo.PurchaseOrderDetail
(
    purchaseOrderID int NOT NULL,
    lineNumber smallint NOT NULL,
    productID int NULL,
    unitPrice money NULL,
    orderQty smallint NULL,
    receivedQty float NULL,
    rejectedQty float NULL,
    dueDate datetime NULL
);
```

从上述代码可以看出, PurchaseOrderDetail 表中包含多个列。在定义数据列时, 数据列之间需要使用逗号(,)分隔。

3.3 维护表

创建数据表完成后, 对数据表的维护也很重要。例如, 在表中增加新的数据列、删除不需要的数据列, 或者更改表的名称等。

3.3.1 管理表中的列

管理表中的列是指列的添加和删除, 用户可以通过两种方式添加或删除表中的列, 即使用图形界面和使用 Transact-SQL 语句。

1. 通过图形界面操作

在图形界面中添加或删除列, 可使用下列步骤。

(1) 右键单击【对象资源管理器】窗格中要操作的数据库中的表。

(2) 在弹出的快捷菜单中选择【设计】命令打开设计器窗口。

(3) 如果要在表中添加列, 在所有列的后面设置要添加的列名称、类型和是否为 Null 值即可; 如果要删除某列, 只需要右键单击该列, 然后在弹出的快捷菜单中选择【删除列】命令即可, 如图 3-9 所示。

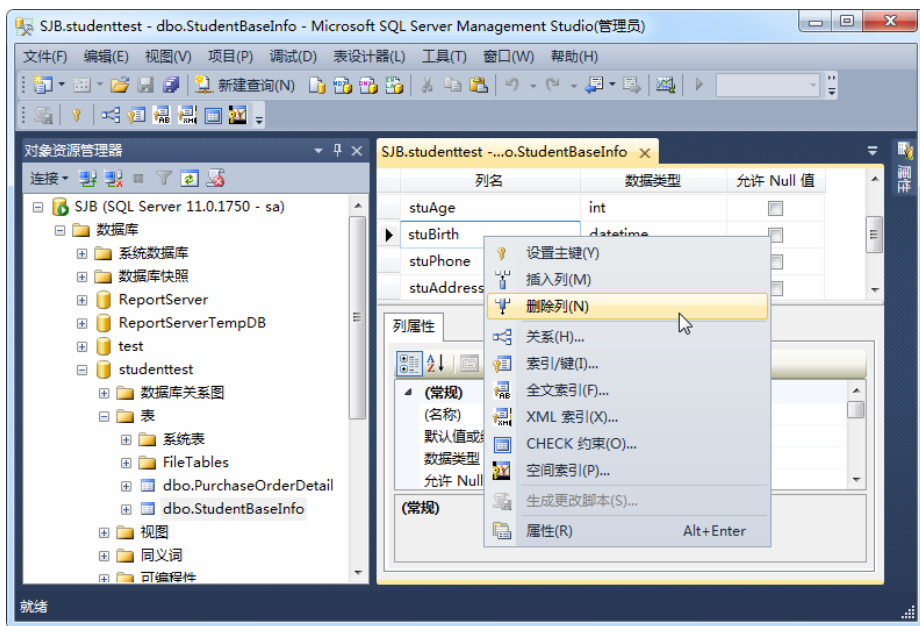


图 3-9 添加或删除列的操作界面

(4) 设置完成后, 保存并关闭设计器窗口。

2. Transact-SQL 语句操作

使用 ALTER TABLE 语句也可以完成添加或删除列的操作。

【范例 3】

如下代码通过 ALTER TABLE 语句在 studenttest 数据库的 StudentBaseInfo 表中添加 stuIntro 列：

```
USE studenttest;
GO
ALTER TABLE StudentBaseInfo ADD stuIntro text NULL;
```

从上述代码可以看出，在 StudentBaseInfo 表中添加 stuIntro 列时，该列的数据类型为 text，并且允许该列有 NULL 值。

可以一次使用 ALTER TABLE 语句添加多个列，在添加多个列时，多个列之间通过逗号 (,) 进行分隔。

【范例 4】

通过 ALTER TABLE 语句添加列时需要用到 ADD 关键字，但是通过该语句删除列时需要使用 DROP COLUMN 关键字。删除 stuIntro 列的代码如下：

```
USE studenttest;
GO
ALTER TABLE StudentBaseInfo DROP COLUMN stuIntro;
```



在删除列时，如果列具有以下特征，则列不能删除：用于 CHECK、FOREIGN KEY、UNIQUE 或 PRIMARY KEY 约束；用于索引；与 DEFAULT 定义管理或绑定到某一默认对象；绑定到规则；用作表的全文键；已注册支持全文。

ALTER TABLE 语句不仅能够实现添加列和删除列的操作，也可以对列的属性进行修改。例如，重新设置列名、数据类型、长度、是否允许为空、描述、默认值、精度等，还可以设置和取消一个列的主键约束。

3. 重命名列

重命名列有两种方式：一种是通过图形界面工具；另一种是使用 Transact-SQL 语句。

通过图形界面对列进行重命名操作时，需要先找到重命名的列，然后右键单击该列，在弹出的快捷菜单中选择【重命名】命令，如图 3-10 所示。另外，用户也可以直接打开表的设计器窗口，在窗口中进行更改，更改完成后保存即可。

通过 Transact-SQL 语句重命名列时，可以使用系统存储过程 sp_rename。该存储过程表示在当前数据库中更改用户创建对象的名称，该对象可以是表、索引、列、别名数据类型或用户定义类型。基本语法如下：

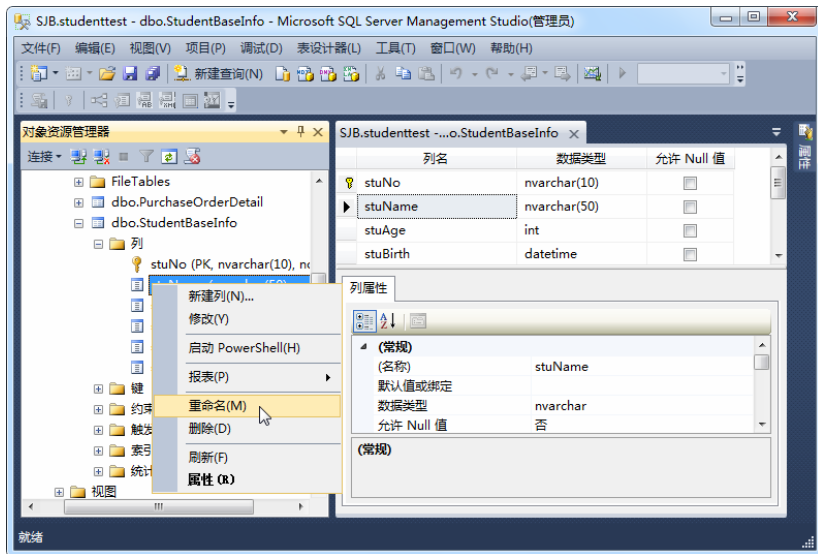


图 3-10 重命名列

```
sp_rename [ @objname = ] 'object_name' , [ @newname = ] 'new_name' [ ,
[ @objtype = ] 'object_type' ]
```

上述语法的说明如下。

(1) [@objname=]'object_name': 用户对象或数据类型的当前限定或非限定名称。如果要重命名的对象是表中的列, 则 object_name 的形式必须是 table.column 或 schema.table.column; 如果要重命名的对象是索引, 则 object_name 的形式必须是 table.index 或 schema.table.index; 如果要重命名的对象是约束, 则 object_name 的形式必须是 schema.constraint。

(2) [@newname=]'new_name': 指定对象的新名称。

(3) [@objtype=]'object_type': 要重命名的对象的类型。object_type 的数据类型是 varchar(13), 默认值为 NULL, 且可以是 COLUMN、DATABASE、INDEX、Object、STATISTICS、USERDATATYPE 的值之一。

【范例 5】

sp_rename 返回的值为 0 或者非零数字, 返回 0 时表示执行成功, 否则表示执行失败。下面的代码将 StudentBaseInfo 表中的 stuAddress 列重命名为 stuNewAddress:

```
USE studenttest;
GO
EXEC sp_rename 'StudentBaseInfo.stuAddress', 'stuNewAddress', 'COLUMN';
GO
```

3.3.2 修改表名

当用户向数据库中添加表之后, 发现表的名称出现了一个错误, 这时需要修改表名。

修改表名可以通过图形界面工具，也可以通过 Transact-SQL 语句。通过图形界面的方式修改表名很简单，一般步骤是：依次展开【对象资源管理器】|【数据库】|studenttest|【表】节点，右键单击【表】节点下要修改的表，在弹出的快捷菜单中选择【重命名】命令，然后进行更改即可。

通过 Transact-SQL 语句对表进行重命名时，也需要使用 sp_rename 存储过程。如下代码将 StudentBaseInfo 表重命名为 StudentInfo：

```
USE studenttest;
GO
EXEC sp_rename 'StudentBaseInfo', 'StudentInfo';
GO
```



在对表进行修改时，首先要查看该表是否和其他表存在依赖关系，如果存在依赖关系，那么应该解除该表的依赖关系后再对表进行修改操作，否则将有可能导致其他表出错。

3.3.3 删除表

用户在 SQL Server 2012 中可以通过使用图形界面或 Transact-SQL 语句从数据库中删除表。通过图形界面删除表的一般步骤是：展开【数据库】节点下的【表】节点，右键单击要删除的表，在弹出的快捷菜单中选择【删除】命令即可。

通过 Transact-SQL 删除表时需要使用 DROP TABLE 语句。DROP TABLE 用于删除一个或多个表定义以及这些表中的所有数据、索引、触发器、约束和权限规范。任何引用已删除表或存储过程时都必须使用 DROP VIEW 或 DROP PROCEDURE 显式删除。DROP TABLE 语句的基本语法如下：

```
DROP TABLE [ database_name . [ schema_name ] . | schema_name . ] table_name
[ , ...n ] [ ; ]
```

其中，database_name 表示要在其中创建表的数据库的名称，schema_name 表示表所属架构的名称，table_name 表示要删除的表的名称。

【范例 6】

如下代码通过 DROP TABLE 删除 PurchaseOrderDetail 表：

```
DROP TABLE dbo.PurchaseOrderDetail;
```

通过 DROP TABLE 语句删除表时，存在以下的限制和约束。

(1) 不能删除被 FOREIGN KEY 约束引用的表。必须先删除引用 FOREIGN KEY 的约束或引用表。如果要在同一个 DROP TABLE 语句中删除引用表以及包含主键的表，则必须先列出引用表。

(2) 删除表时，表的规则或默认值将被解除绑定，与该表关联的任何约束或触发器将被自动删除。如果要重新创建表，则必须重新绑定相应的规则和默认值，重新创建某

- 些触发器，并添加所有必需的约束。
- (3) 如果删除的表包含带有 FILESTREAM 属性的 varbinary(max)列，则不会删除在文件系统中存储的任何数据。
 - (4) 不应在同一个批处理中对同一个表执行 DROP TABLE 和 CREATE TABLE，否则可能出现意外错误。
 - (5) 任何引用已删除表的视图或存储过程都必须显式删除或修改，以便删除对该表的引用。

3.3.4 查看表定义

用户可以通过图形界面或 Transact-SQL 语句查看某个表的属性。通过图形界面显示表属性时，需要右键单击选中的表，然后在弹出的快捷菜单中选择【属性】命令，弹出【表属性】对话框，如图 3-11 所示。在该对话框中，单击不同的选项页可查看表的基本信息、权限、扩展属性以及存储等内容。

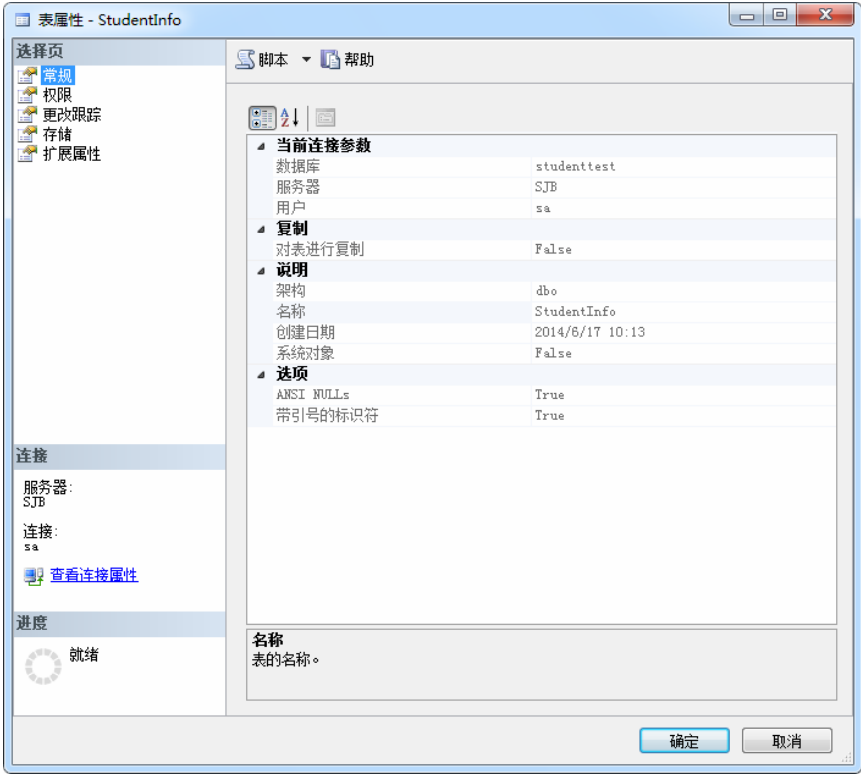


图 3-11 查看表属性

通过 Transact-SQL 查看表属性时，可以使用 sys.tables 目录视图。sys.tables 为每个表的对象返回一行，当前仅用于 sys.objects.type=U 的表对象。如图 3-12 所示的查询当前数据库中存在的表，并返回一系列与表有关的属性。

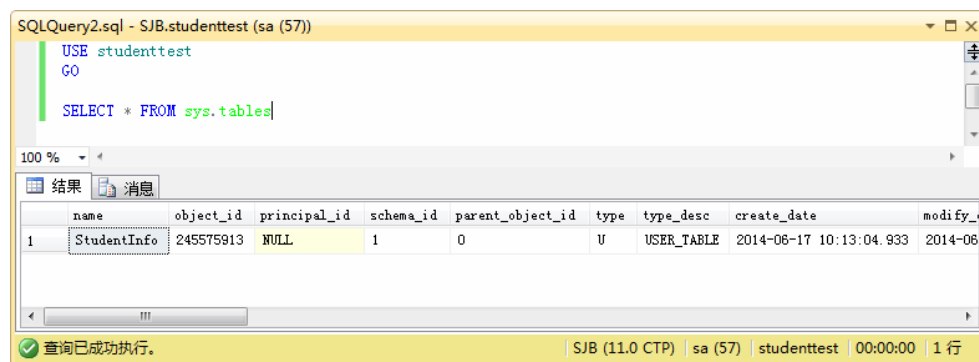


图 3-12 sys.tables 的使用

3.4 指定列数据类型

SQL Server 数据库使用不同的数据类型存储数据，并且要求所有的列中的数值都有相同的数据类型（除非指定 SQL_VARIANT 数据类型的值）。数据类型是一种属性，用于指定对象可保存的数据的类型。SQL Server 提供系统数据类型集，该类型集定义了可与 SQL Server 一起使用的所有数据类型。

本节介绍指定列时 SQL Server 提供的数据类型，如数字类型、日期和时间类型、字符串类型以及 Unicode 字符串类型等。

3.4.1 数字类型

数字类型是指与数字有关的类型，大体可以将数字类型分为精确数字类型和近似数字类型。精确数字类型能够精确到准确的数字，SQL Server 中的精确数字类型有 bigint、bit、decimal、int、money、numeric、smallint、smallmoney 和 tinyint，近似数字类型有 float 和 real 两种。

1. bigint、int、smallint、tinyint 类型

在这 4 种类型中，int 类型最经常被用到，bigint 数据类型用于整数值可能超过 int 数据类型支持范围的情况。在数据类型优先次序表中，bigint 介于 smallmoney 和 int 之间。如表 3-2 所示是对 bigint、int、smallint 和 tinyint 类型进行的简单说明。

表 3-2 bigint、int、smallint 和 tinyint 类型

数据类型	范围	存储
bigint	$-2^{63}(-9\ 223\ 372\ 036\ 854\ 775\ 808) \sim 2^{63}-1(9\ 223\ 372\ 036\ 854\ 775\ 807)$	8 字节
int	$-2^{31}(-2\ 147\ 483\ 648) \sim 2^{31}-1(2\ 147\ 483\ 647)$	4 字节
smallint	$-2^{15}(-32\ 768) \sim 2^{15}-1(32\ 767)$	2 字节
tinyint	0~255	1 字节

2. bit 类型

bit 是可以取值为 1、0 或 NULL 的 integer 数据类型。SQL Server 数据库引擎可优化 bit 列的存储。如果表中的列为 8bit 或者更少, 则这些列作为 1 个字节存储; 如果列为 9~16bit, 则这些列作为 2 个字节存储, 以此类推。

字符串值 True 或者 False 可转换为 bit 值, True 将转换为 1, False 将转换为 0。转换时 bit 会将任何非零值转换为 1。

3. decimal 和 numeric 类型

decimal 和 numeric 是带固定精度和小数位数的数值数据类型。使用最大精度时, 有效值的范围为 $-10^{38}+1 \sim 10^{38}-1$ 。语法如下:

```
decimal [ (p[,s] ) ] 和 numeric [ (p[,s] ) ]
```

其中 p 表示精度, 最多可以存储的十进制数字的总位数, 包括小数点左边和右边的位数。该精度必须是从 1 到最大精度 38 之间的值。默认精度为 18。 s 表示小数位数, 小数点右边可以存储的十进制数字的位数, 从 p 中减去此数字可确定小数点左边的最大位数, 小数位数必须是 0 到 p 之间的值。仅在指定精度后才可以指定小数位数, 默认的小数位数为 0。因此 $0 \leq s \leq p$ 。

4. money 和 smallmoney 类型

money 和 smallmoney 是代表货币或货币值的数据类型, 这两种数据类型精确到它们所代表的货币单位的万分之一。money 数据类型存储 8 字节, 范围是 -922 337 203 685 477.5808~922 337 203 685 477.5807。smallmoney 数据类型存储 4 字节, 范围是 -214 748.3648 到 214 748.3647。

5. float 和 real 类型

float 和 real 是用于表示浮点数值数据的大致数值数据类型。浮点数据为近似值, 因此, 并非数据类型范围内的所有值都能精确地表示。float 类型的语法如下:

```
float [ (n) ]
```

其中 n 为用于存储 float 数值尾数的位数 (以科学记数法表示), 因此可以确定精度和存储大小。如果指定了 n , 则它必须是介于 1~53 之间的某个值, 默认值为 5。当 n 的取值在 1~24 之间时, 精度为 7 位数, 存储大小为 4 字节; 当 n 的取值在 25~53 之间时, 精度为 15 位数, 存储大小为 8 字节。

3.4.2 日期和时间类型

SQL Server 2012 中支持 6 种日期和时间类型, 它们的简单说明如下。

1. date 类型

`date` 数据类型定义一个日期,支持的日期范围从 0001-01-01~9999-12-31,即公元元年 1 月 1 日到公元 9999 年 12 月 31 日,默认值为 1900-01-01。存储 `date` 数据类型磁盘开销需三个字节,如果只需要存储日期值而没有时间,使用 `date` 可以比 `smalldatetime` 节省一个字节的磁盘空间。

`date` 数据类型的默认字符串文字格式是 YYYY-MM-DD,其中 YYYY 是表示年份的 4 位数字,范围是 0001~9999;MM 是表示指定年份中的月份的两位数字,范围是 01~12;DD 是表示指定月份中的某一天的两位数字,范围是 01~31。

2. datetime 类型

`datetime` 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结合的日期。支持的日期范围是 1753 年 1 月 1 日~9999 年 12 月 31 日,时间范围是 00:00:00~23:59:59.997,默认值为 1900-01-01 00:00:00。

`datetime` 数据类型的一般格式是 YYYY-MM-DD hh:mm:ss。其中 YYYY 是表示年份的 4 位数字,范围为 1753~9999;MM 是表示指定年份中的月份的两位数字,范围是 01~12;DD 是表示指定月份中的某一天的两位数字,范围是 01~31;hh 是表示小时的两位数字,范围是 00~23;mm 是表示分钟的两位数字,范围是 00~59;ss 是表示秒钟的两位数字,范围是 00~59。

3. datetime2 类型

`datetime2` 类型定义结合了 24 小时制时间的日期,可将 `datetime2` 视作现有 `datetime` 类型的扩展,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` 类型指定的日期范围是 0001-01-01~9999-12-31,即公元元年 1 月 1 日到公元 9999 年 12 月 31 日,时间范围是指 00:00:00~23:59:59.9999999。

4. datetimeoffset 类型

`datetimeoffset` 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结合的日期。该类型要求存储的日期和时间(24 小时制)是与时区一致的,时间部分能够支持高达 100 ns 的精确度。

5. smalldatetime 类型

`smalldatetime` 类型定义结合了一天中的时间的日期,该时间为 24 小时制,秒始终为零,并且不带秒小数部分。`smalldatetime` 类型的默认值为 1900-01-01 00:00:00,该类型定义的日期范围是 1900-01-01~2079-06-06,即 1900 年 1 月 1 日~2079 年 6 月 6 日,时间范围是 00:00:00~23:59:59,如 2007-05-09 23:59:59 将舍入为 2007-05-10 00:00:00。

6. time 类型

`time` 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。`time` 类

型的范围是 00:00:00.0000000~23:59:59.9999999, 它支持高达 100ns 的精确度。如果想存储一个特定的时间信息而不涉及具体的日期, time 数据类型非常有用。

3.4.3 字符串类型

在 SQL Server 2012 中, 提供了 char、varchar 和 text 三种字符串类型。

1. char 类型

char 类型表示固定长度, 非 Unicode 字符串数据。基本形式如下:

```
char [ ( n ) ]
```

其中, n 用于定义字符串长度, 并且它必须为 1~8 000 之间的值。存储大小为 n 字节, 在列数据项的大小一致时使用。

2. varchar 类型

varchar 类型表示可变长度, 非 Unicode 字符串数据。基本形式如下:

```
varchar [ ( n | max ) ]
```

其中, n 用于定义字符串长度, 并且它可以为 1~8 000 之间的值。max 指示最大存储大小是 $2^{31}-1$ 个字节, 即 2GB。存储大小为所输入数据的实际长度+2 个字节。

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, 则使用 varchar(max)。

3. text 类型

text 类型为服务器代码页中长度可变的非 Unicode 数据, 字符串最大长度为 $2^{31}-1$ (2 147 483 647) 个字节。当服务器代码页使用双字节字符时, 存储仍是 2 147 483 647 字节。根据字符串, 存储大小可能小于 2 147 483 647 字节。

3.4.4 Unicode 字符串类型

Unicode 是一种在计算机上使用的字符编码。在 SQL Server 2012 中, 提供了 nchar、nvarchar 和 ntext 三种 Unicode 字符串类型。

1. nchar 类型

nchar 类型的长度固定, 在列数据项的大小可能相同时使用。基本形式如下:

```
nchar[ ( n ) ]
```

其中, n 的值必须在 1~4 000 之间。如果没有在数据定义或变量声明语句中指定 n , 则默认值长度为 1。

2. nvarchar 类型

nvarchar 类型存储可变长度 Unicode 字符数据。在列数据项的大小可能差异很大时使用。基本形式如下：

```
nvarchar[ ( n | max ) ]
```

其中, n 的值在 1~4 000 之间。如果没有在数据定义或变量声明语句中指定 n , 则默认值长度为 1。

3. ntext 类型

ntext 类型存储长度可变的 Unicode 数据, 字符串最大长度为 $2^{30}-1$ (1 073 741 823) 个字节。存储大小是所输入字符串长度的两倍 (以字节为单位)。

3.4.5 二进制字符串

SQL Server 2012 中支持二进制类型, 二进制类型用于存储二进制数据。下面介绍 binary、varbinary 和 image 三种二进制字符串类型。

1. binary 类型

如果列数据项的大小一致时, 可以使用 binary 类型。binary 类型表示长度为 n 字节的固定长度二进制数据, 其中 n 是指从 1~8 000 的值, 存储大小为 n 字节。如果 n 没有指定, 则默认长度为 1。基本形式如下:

```
binary [ ( n ) ]
```

2. varbinary 类型

varbinary 存储可变长度的二进制数据。基本形式如下:

```
varbinary [ ( n | max ) ]
```

其中, n 的取值范围为 1~8 000 之间, max 指示最大存储大小是 $2^{31}-1$ 个字节。存储大小为所输入数据的实际长度+2 个字节。如果 n 没有指定, 则默认长度为 1。

如果列数据项的大小差异相当大, 则使用 varbinary 类型; 当列数据条目超出 8 000 字节时, 则使用 varbinary(max) 类型。

3. image 类型

image 代表长度可变的二进制数据, 可以存储 $0\sim 2^{31}-1$ (2 147 483 647) 个字节。

3.4.6 其他数据类型

除了前面介绍的几种数据类型外, SQL Server 2012 中还提供了其他的数据类型, 如 cursor、timestamp、hierarchyid 和 uniqueidentifier 等。

1. cursor 类型

cursor 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，对于 CREATE TABLE 语句中的列，不能使用 cursor 数据类型。

2. timestamp 类型

timestamp 类型表示时间戳，一般用作给表行加版本戳的机制。timestamp 值是一个二进制数值，表明数据库中的数据修改发生的相对顺序。

3. hierarchyid 类型

hierarchyid 是一种长度可变的系统数据类型。使用它来表示层次结构中的位置，类型为 hierarchyid 的列不会自动表示数。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。

4. uniqueidentifier 类型

uniqueidentifier 可以存储 16 字节的二进制值，其作用与全局唯一标识符（即 GUID）一样。uniqueidentifier 列的 GUID 值可以在 Transact-SQL 语句、批处理或脚本中调用 NEWID() 函数获取。

5. sql_variant 类型

sql_variant 也是一种数据类型，用于存储 SQL Server 支持的各种数据类型的值。sql_variant 可以用在列、参数、变量和用户定义函数的返回值中，它使这些数据库对象能够支持其他数据类型的值。

6. xml 类型

xml 是一种用于存储 XML 数据的数据类型，可以在列中或者 xml 类型的变量中存储 xml 实例。基本语法如下：

```
xml ( [ CONTENT | DOCUMENT ] xml_schema_collection )
```

其中，CONTENT 将 xml 实例限制为格式正确的 XML 片段，XML 数据的顶层可包含多个零或者多个元素，还允许在顶层使用文本节点，这是默认行为。DOCUMENT 将 xml 实例限制为格式正确的 XML 文档，XML 数据必须且只能有一个根元素，不允许在顶层使用文本节点。xml_schema_collection 是 XML 架构集合的名称，如果要创建类型化的 xml 列或变量，可以选择指定的 XML 架构集合名称。

7. table 类型

table 是一种特殊的数据类型，可用于存储结果集以进行后续处理。table 主要用于临时存储一组作为表值函数的结果集返回的行，可将函数和变量声明为 table 类型。table

变量可用于函数、存储过程和批处理中，如果要声明 table 类型的变量，需要使用 DECLARE @local_variable。

8. 空间类型

SQL Server 2012 支持 geography 和 geometry 两种空间类型，分别说明如下。

(1) geography 类型：表示地理空间类型，它在 SQL Server 中作为公共语言运行时数据类型实现。该类型表示圆形地球坐标系中的数据，用于存储诸如 GPS 纬度和经度坐标之类的椭球体数据。

(2) geometry 类型：表示平面空间数据类型，它在 SQL Server 中作为公共语言运行时数据类型实现。该类型表示平面坐标系中的数据。

3.4.7 用户自定义数据类型

所谓的用户自定义数据类型，是指用户基于系统的数据类型而设计并实现的数据类型。创建用户定义的数据类型时必须提供数据类型的名称、所基于的系统数据类型和是否允许为空三个参数。

在 SQL Server 2012 中，用户通常可以通过图形界面和 Transact-SQL 语句来自定义数据类型。

1. 通过图形界面创建类型

通过图形界面创建数据类型的方法很简单，下面通过范例 7 进行说明。

【范例 7】

通过图形界面自定义数据类型 mysc，它基于 float 系统数据类型，并且不允许为空。实现步骤如下。

(1) 在【对象资源管理器】窗格找到某一个数据库下的【可编程性】节点，并展开该节点。

(2) 右键单击【可编程性】|【类型】节点，如图 3-13 所示。

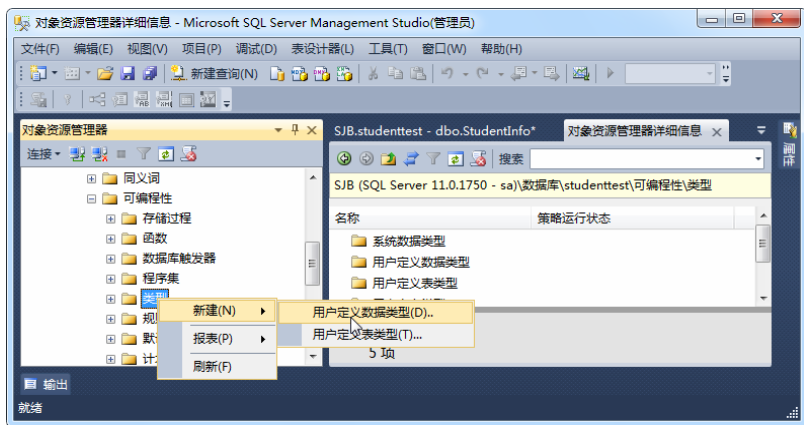


图 3-13 用户定义数据类型

(3) 在图 3-13 弹出的快捷菜单中选择【新建】|【用户定义数据类型】命令，弹出如图 3-14 所示的对话框。在该对话框中的【名称】文本框中输入 mysc，然后选择数据类型为 float，且保证文本“允许 NULL 值”之前的复选框不被选中。

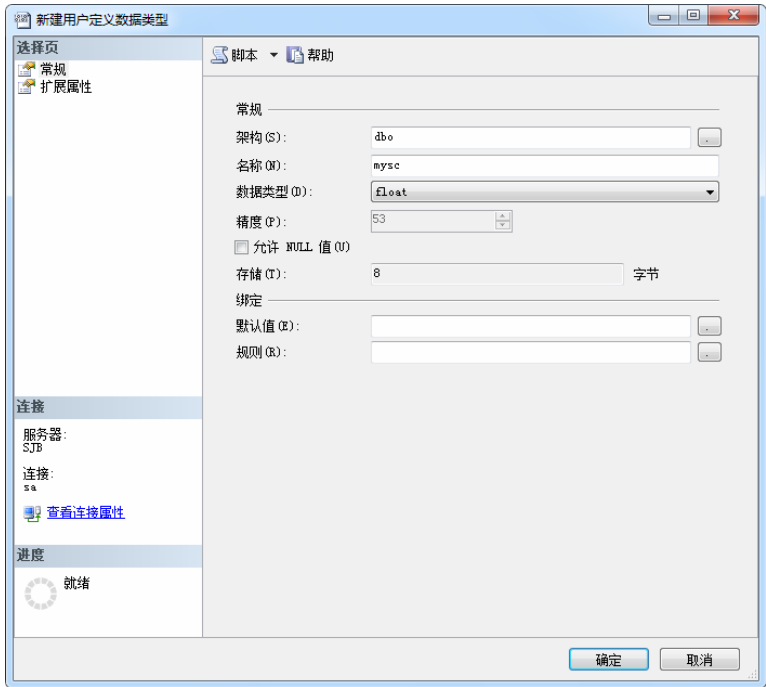


图 3-14 新建用户定义数据类型

(4) 设置完成后单击【确定】按钮，添加完成后可以在【可编程性】节点下找到该类型。

2. 通过 Transact-SQL 语句创建类型

通过 Transact-SQL 语句自定义数据类型时，需要调用 sp_addtype 存储过程。基本语法如下：

```
sp_addtype [ @typename = ] type, [ @phystype = ] system_data_type [ ,  
[ @nulltype = ] 'null_type' ] ;
```

上述语法的说明如下。

(1) [@typename =] type: 自定义数据类型的名称，该名称必须遵循标识符规则，并且在每个数据库中是唯一的。type 的数据类型为 sysname，无默认值。

(2) [@phystype =] system_data_type: 自定义数据类型所基于的物理数据类型或 SQL Server 提供的数据类型。system_data_type 的数据类型为 sysname，无默认值。

(3) [@nulltype =] 'null_type': 指定自定义数据类型处理空值的方式。null_type 的数据类型为 varchar(8)，默认值为 NULL，并且必须用单引号引起来 ('NULL'、'NOT NULL' 或 'NONULL')。

【范例 8】

使用 `sp_addtype` 存储过程定义基于 `float` 系统数据类型、且不允许有空值的 `mysc` 数据类型。代码如下：

```
USE studenttest;
GO
EXEC sp_addtype mysc, 'float', 'NOT NULL';
```

如果要删除自定义的数据类型，用户可以使用 `sp_droptype` 存储过程来完成。基本语法如下：

```
sp_droptype [ @typename = ] 'type'
```

其中，`[@typename=]'type'` 表示用户所拥有的自定义数据类型的名称，`type` 的数据类型为 `sysname`，无默认值。

【范例 9】

如下代码使用 `sp_droptype` 存储过程删除自定义的 `mysc` 数据类型：

```
USE studenttest;
GO
EXEC sp_droptype mysc;
```

3.4.8 数据类型的优先级

当两个不同数据类型的表达式用运算符组合后，数据类型优先级规则指定将优先级较低的数据类型转换为优先级较高的数据类型。如果此转换不是所支持的隐式转换，则会返回错误。当两个操作数表达式具有相同的数据类型时，运算的结果便为该数据类型。

在 SQL Server 的数据类型中，数据类型的优先级是：用户自定义数据类型（最高）> `sql_variant` > `xml` > `datetimeoffset` > `datetime2` > `datetime` > `smalldatetime` > `date` > `time` > `float` > `real` > `decimal` > `money` > `smallmoney` > `bigint` > `int` > `smallint` > `tinyint` > `bit` > `ntext` > `text` > `image` > `timestamp` > `uniqueidentifier` > `nvarchar`（包括 `nvarchar(max)`）> `nchar` > `varchar`（包括 `varchar(max)`）> `char` > `varbinary`（包括 `varbinary(max)`）> `binary`（最低）。

3.5 约束类型

约束是 SQL Server 强制实行的应用规则，建立和使用约束的目的是保证数据的完整性。约束能够限制用户存放到表中数据的格式和可能值，它作为数据库定义的一部分在 `CREATE TABLE` 语句中声明，因此又称作声明完整性约束。

约束独立于表结构，可以在不改变表结构的情况下，通过 `ALTER TABLE` 语句来添加或者删除。在删除一个表时，该表所带的所有约束定义也被随之删除。

3.5.1 主键约束

在表中经常有一列或多列的组合，其值能够唯一标识表中的每一行。这样的一列或多列称为表的主键，一个表只能包含一个主键约束，通过它可以强制表的实体完整性。由于主键约束可保证数据的唯一性，因此经常对标识列定义这种约束。

如果为表指定了主键约束，数据库引擎将通过为主键列自动创建唯一索引来强制数据的唯一性。当在查询中使用主键时，此索引还允许对数据进行快速访问。如果对多列定义了主键约束，则一行中的值可能会重复，但是来自主键约束定义中所有列的值的任何组合必须是唯一的。

1. 创建主键约束

创建主键将自动创建相应的唯一索引、聚集索引或非聚集索引。用户可以图形界面和 Transact-SQL 语句两种方式创建主键约束。在范例 1 中通过图形界面创建表时已经演示了主键约束的创建，因此这里不再详细说明。

通过 Transact-SQL 语句定义主键时可以分为两种方式：一种是在创建表时创建，需要使用 PRIMARY KEY；另一种是修改表中的主键，需要使用 ALTER TABLE。

【范例 10】

创建名称为 Persons 的数据表，该表包含 personId、personName 和 personAddress 三个列，在创建时指定 personId 列为主键。代码如下：

```
CREATE TABLE Persons
(
    personId int NOT NULL PRIMARY KEY,
    personName varchar(50) NOT NULL,
    personAddress varchar(100)
)
```

【范例 11】

除了范例 10 的创建方式外，还可以在表的字段列创建完成后通过 CONSTRAINT 指定主键。如下代码等价于范例 10 的代码：

```
CREATE TABLE Persons
(
    personId int NOT NULL,
    personName varchar(50) NOT NULL,
    personAddress varchar(100),
    CONSTRAINT pk_PersonID PRIMARY KEY (personID)
)
```

【范例 12】

通过 ALTER TABLE 语句为现有的表添加主键。如下代码等价于范例 10 和范例 11

的代码:

```
CREATE TABLE Persons
(
    personId int NOT NULL,
    personName varchar(50) NOT NULL,
    personAddress varchar(100)
)
GO
ALTER TABLE Persons ADD CONSTRAINT pk_PersonID PRIMARY KEY (personId)
```

注意

如果使用 ALTER TABLE 语句添加主键, 那么在表首次创建时必须把主键列声明为不包含 NULL 值。

2. 修改主键约束

用户可以通过图形界面修改主键约束, 首先打开要修改主键的表的设计器, 然后右键单击主键列, 在弹出的快捷菜单中选择【索引/键】命令, 弹出【索引/键】对话框, 如图 3-15 所示。在该对话框中, 从左侧【选定的主/唯一键或索引】列表中选中主键索引, 然后在右侧【列】选项后单击按钮, 弹出【索引列】对话框, 如图 3-16 所示, 在该对话框中修改主键。修改完成后单击对话框中的【确定】按钮。

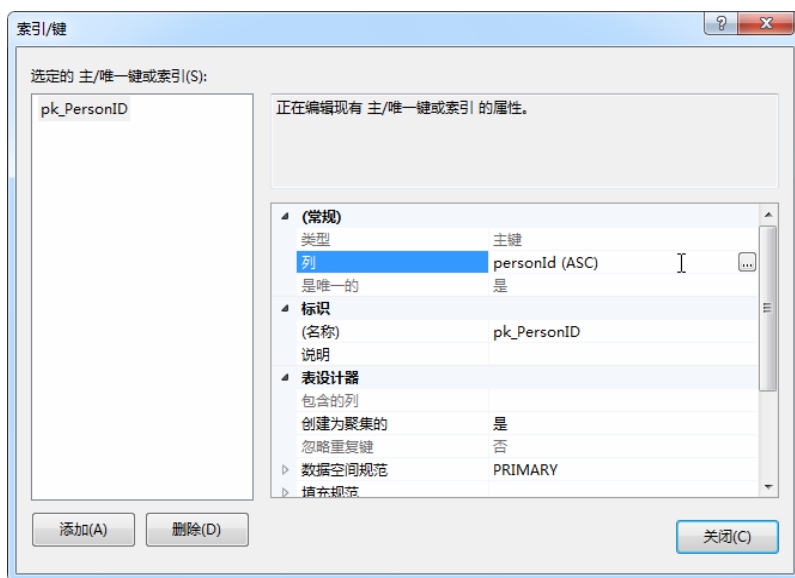


图 3-15 修改主键约束 1

如果要使用 Transact-SQL 修改主键约束, 那么必须先删除现有的主键约束, 然后再用 PRIMARY KEY 重新创建该约束。

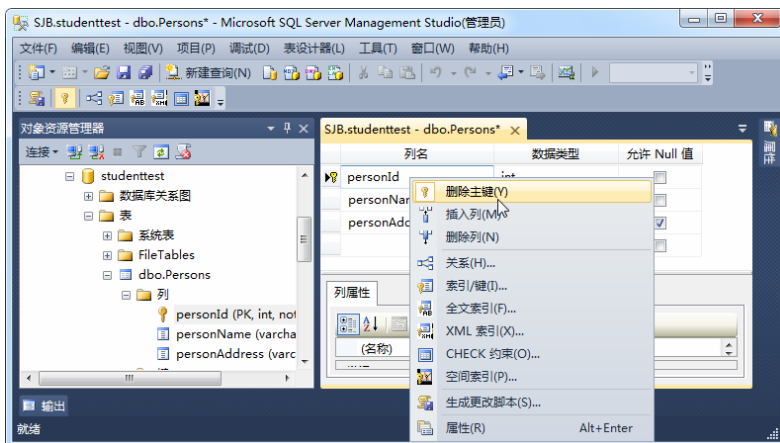


图 3-18 在设计器中删除主键

通过 Transact-SQL 语句删除主键时需要使用 ALTER TABLE 语句,在该语句中还需要用到 DROP CONSTRAINT。如下代码删除 Persons 表中添加的主键约束:

```
ALTER TABLE Persons DROP CONSTRAINT pk_PersonID
```

3.5.2 自动增长标识

SQL Server 为自动进行顺序编号引入了自动编号的 IDENTITY 属性,具有 IDENTITY 属性的列称为标识列,其取值称为标识值,具有以下特点。

(1) IDENTITY 列的数据类型只能为 tinyint、smallint、int、bigint、numeric 和 decimal。当为 numeric 和 decimal 类型时,不允许有小数位。

(2) 当用户向表中插入一行新的记录时,不必也不能向具有 IDENTITY 属性的列输入数据,系统将自动在该列添加一个按规定间隔递增或递减的数据。

(3) 每个表最多有一列具有 IDENTITY 属性,且该列不能为空,不允许具有默认值,也不能由用户更新。

IDENTITY 属性的语法如下:

```
IDENTITY [ (seed , increment) ]
```

其中,seed 表示加载到表中的第一个行所使用的值,increment 表示与前一个加载的行的标识值相加的增量值。使用 IDENTITY 属性时,必须同时指定种子和增量,或者二者都不指定。如果两者都未指定,则取默认值(1,1)。

【范例 13】

通过 CREATE TABLE 语句创建 Persons 表,在创建表时通过 IDENTITY(1, 1)设置自动增长列和主键。代码如下:

```
CREATE TABLE Persons
(
    personId int IDENTITY(1,1) PRIMARY KEY NOT NULL,
```

```
personName varchar(50) NOT NULL,  
personAddress varchar(100)  
)
```

**提示**

除了在创建时使用 IDENTITY 指定自动增长的种子和增量外，还可以通过 ALTER TABLE 设置自动增量，也可以在表的设计器窗口中设置列属性（如图 3-10 所示），这里以及后面的小节只介绍常用的操作方式，不再对每一种方式进行介绍。

3.5.3 唯一性约束

一个表只能有一个主键，如果有多列或多个列组合需要确保数据唯一性，这时需要通过 UNIQUE 进行定义。通常情况下，也将唯一性约束称为唯一约束。唯一性约束指定的列可以有 NULL 属性。由于主键值是具有唯一性的，因此主键列不能再设定唯一性约束。

尽管 UNIQUE 约束和 PRIMARY KEY 约束都强制唯一性，但如果要强制一列或多列组合（不是主键）的唯一性时应使用 UNIQUE 约束而不是 PRIMARY KEY 约束。两者的区别如下。

（1）可以对一个表定义多个 UNIQUE 约束，但只能定义一个 PRIMARY KEY 约束。

（2）UNIQUE 约束允许 NULL 值，这一点与 PRIMARY KEY 约束不同。不过，当和参与 UNIQUE 约束的任何值一起使用时，每列只允许一个空值。

（3）FOREIGN KEY 约束可以引用 UNIQUE 约束。

【范例 14】

在创建 Persons 表时，指定该表中 personEmail 列的值是唯一的。代码如下：

```
CREATE TABLE Persons  
(  
    personId int IDENTITY(1,1) PRIMARY KEY NOT NULL,  
    personName varchar(50) NOT NULL,  
    personAddress varchar(100),  
    personEmail nvarchar(50),  
    CONSTRAINT AK_OneEmail UNIQUE(personEmail)  
)  
GO
```

如果要删除唯一性约束，可以使用 ALTER TABLE 语句。代码如下：

```
ALTER TABLE Persons DROP CONSTRAINT AK_OneEmail;
```

根据上述代码以及删除主键约束的代码可以总结出删除约束的一般语法，内容如下：

```
ALTER TABLE 表名 DROP CONSTRAINT 约束名称;
```

这种删除约束的语法正好与 ALTER TABLE 添加约束的语法对应，如下为添加唯一

性约束的语法:

```
ALTER TABLE 表名 ADD CONSTRAINT 约束名称 UNIQUE (字段列);
```

其中,UNIQUE 表示创建唯一性约束,如果要创建主键约束,则使用 PRIMARY KEY;如果要创建检查约束,可以使用 CHECK。

3.5.4 空与非空约束

列的空性决定了在表中该列上是否可以使用空值。出现 NULL 通常表示值未知或未定义。空值(或 NULL)不同于零、空白或者长度为零的字符串。如果使用 NULL 约束,需要注意以下几点。

(1) 如果插入了一行,但没有为允许 NULL 值的列包含任何值,除非存在 DEFAULT 定义或 DEFAULT 对象,否则数据库引擎将提供 NULL 值。

(2) 用 NULL 关键字定义的列接受用户输入的 NULL 显式输入,不论它是何种数据类型,或者是否有默认值与之关联。

(3) NULL 值不应该放在引号内,否则会被解释为字符串 NULL 而不是空值。

指定某一列不允许为空值有助于维护数据的完整性,因为这样可以确保行中的列永远包含数据。如果不允许为空,用户向表中输入数据时必须在列中输入一个值,否则数据库将不接受该表行。

在前面的范例创建表时,已经使用到了 NOT NULL,它表示该列不能为空,如果不设置,则该列默认为空。以范例 14 创建 Persons 表为例,personId、personName 列不允许为空值,且 personId 列为主键,主键的值自动增长;personAddress 列和 personEmail 列允许有空值。

3.5.5 默认值约束

在向表中插入数据时,如果没有指定某一列字段的数值,则该字段的数据存在以下三种情况。

- (1) 如果该字段定义有默认值,则系统将默认值插入字段。
- (2) 如果该字段定义没有默认值,但允许为空,则插入空值。
- (3) 如果该字段定义没有默认值,但不允许为空,则报错。

用户在创建表时可以通过 DEFAULT 创建默认值约束,当然也可以通过 ALTER TABLE 更改现有表的 DEFAULT 约束。

【范例 15】

在范例 14 的基础上添加代码,通过 ALTER TABLE 修改 Persons 表的默认值约束。将 personAddress 字段列的默认值指定为“河南省郑州市”。代码如下:

```
ALTER TABLE Persons ADD CONSTRAINT DK_Address DEFAULT('河南省郑州市') for
personAddress;
```

通过默认值约束时需要使用 DEFAULT 关键字,如果使用默认值约束,则需要注意

以下几点。

(1) DEFAULT 约束定义的默认值仅在执行 INSERT 操作插入数据时有效。

(2) 一列最多有一个默认值，其中包括 NULL 值。

(3) 具有 IDENTITY 属性或 timestamp 数据类型属性的列不能使用数据值，text 和 image 类型的列只能以 NULL 为默认值。

3.5.6 检查约束

通过 CHECK 可以设置检查约束，检查约束用来检查用户输入数据的取值是否正确，只有符合约束条件的数据才能输入。在一个表中可以创建多个检查约束，在一个列上也可以创建多个 CHECK 约束，只要它们不相互矛盾。

用户可以在创建表时添加检查约束，也可以更改现有表的检查约束。

【范例 16】

下面创建 Persons 表，分别为表中的 personId 和 personAddress 添加 Check 约束，指定 personId 列的值必须大于 100，personAddress 列的值必须等于“郑州”。代码如下：

```
CREATE TABLE Persons
(
    personId int PRIMARY KEY CHECK(personId>100) NOT NULL,
    personName varchar(50) NOT NULL,
    personAddress varchar(100) CHECK(personAddress='郑州'),
    personEmail nvarchar(50)
)
```

上述代码等价于如下代码：

```
CREATE TABLE Persons
(
    personId int PRIMARY KEY NOT NULL,
    personName varchar(50) NOT NULL,
    personAddress varchar(100),
    personEmail nvarchar(50),
    CONSTRAINT CK_Person CHECK(personId>100 AND personAddress='郑州')
)
```

或者等价于如下代码：

```
CREATE TABLE Persons
(
    personId int PRIMARY KEY NOT NULL,
    personName varchar(50) NOT NULL,
    personAddress varchar(100),
    personEmail nvarchar(50)
)
GO
ALTER TABLE Persons ADD CONSTRAINT PK_Persons CHECK(personId>100 AND
personAddress='郑州');
```

3.5.7 外键约束

外键（FOREIGN KEY）约束保证了数据库各个表中数据的一致性和正确性。外键是用于在两个表中的数据之间建立和加强链接的一系列或多列的组合，可控制可在外键表中存储的数据。在外键引用中，当包含一个表的主键值的一个或多个列被另一个表中的一个或多个列引用时，就在这两个表之间创建链接，这个列就成为第二个表的外键。

例如，销售订单和销售人员之间存在着一种逻辑关系，订单表含有一个指向销售人员表的外键链接。订单表中的 salesOrderPersonId 列与 SalesPerson 表中的主键列相对应，salesOrderPersonId 列是指向 SalesPerson 表的外键。

【范例 17】

下面在创建 SalesOrder 表时，将表中的 salesOrderId 列指定为主键，将表中的 salesOrderPersonId 列指定为外键。代码如下：

```
CREATE TABLE SalesOrder(  
    salesOrderId int NOT NULL,  
    salesOrderNo int NOT NULL,  
    salesOrderPersonId int,  
    PRIMARY KEY(salesOrderId),  
    FOREIGN KEY (salesOrderPersonId) REFERENCES SalesPerson(salePersonId)  
)
```

如果需要指定外键约束的名称，或者为多个列定义外键约束，那么可以使用 CONSTRAINT 定义。

【范例 18】

下面指定外键约束的名称为 FK_OrderPerson，代码如下：

```
CREATE TABLE SalesOrder(  
    salesOrderId int NOT NULL,  
    salesOrderNo int NOT NULL,  
    salesOrderPersonId int,  
    PRIMARY KEY(salesOrderId),  
    CONSTRAINT FK_OrderPerson FOREIGN KEY (salesOrderPersonId) REFERENCES  
    SalesPerson(salePersonId)  
)
```

【范例 19】

用户可以首先创建有关的表，然后为现有的表添加外键约束。代码如下：

```
CREATE TABLE SalesOrder(  
    salesOrderId int PRIMARY KEY NOT NULL,  
    salesOrderNo int NOT NULL,  
    salesOrderPersonId int  
)  
GO
```

```
ALTER TABLE SalesOrder ADD CONSTRAINT FK_OrderPerson FOREIGN KEY  
(salesOrderPersonId) REFERENCES SalesPerson(salePersonId);
```

3.6 实验指导——创建图书管理系统的相关表

一个完整的图书管理系统包含的功能非常强大。在管理系统的后台数据库中，可以包含图书基本信息表、图书类型表、图书借阅表、图书会员表、会员类型表以及归还表等。本节通过完整的代码创建图书表、作者表和图书类型表，并为它们添加关系，实现步骤如下。

(1) 判断当前数据库中是否存在 **bookmanage** 数据库，如果存在则通过 **DROP DATABASE** 语句删除，然后再创建新的数据库。代码如下：

```
IF EXISTS(SELECT * FROM sys.databases WHERE name='bookmanage')  
    DROP DATABASE bookmanage;  
CREATE DATABASE bookmanage;  
GO
```

(2) 通过 **USE** 语句指定使用 **bookmanage** 数据库，代码如下：

```
USE bookmanage;  
GO
```

(3) 向 **bookmanage** 数据库中添加 **BookType** 表，该表包含 **typeId**、**typeName**、**typeParentId** 和 **typeRemark** 4 个列。代码如下：

```
CREATE TABLE BookType  
(  
    typeId int IDENTITY(1,1) PRIMARY KEY NOT NULL,  
    typeName nvarchar(50) NOT NULL,  
    typeParentId int,  
    typeRemark nvarchar(200) DEFAULT '暂无'  
)
```

在上述代码中，将 **typeId** 列设置为主键，自动增长种子和增量都是 1，将 **typeRemark** 列的默认值设置为“暂无”，且 **typeId** 和 **typeName** 列不允许有空值。

(4) 向 **bookmanage** 数据库中添加 **BookAuthor** 表，该表包含 **authorId**、**authorName**、**authorSex**、**authorCountry** 和 **authorIntro** 5 个列。代码如下：

```
CREATE TABLE BookAuthor  
(  
    authorId int IDENTITY(1,1) PRIMARY KEY NOT NULL,  
    authorName nvarchar(50) NOT NULL,  
    authorSex nvarchar(2) DEFAULT('女'),  
    authorCountry nvarchar(20) DEFAULT('不详'),  
    authorIntro text  
)
```

在上述代码中，将 **authorId** 列设置为主键，该列自动增长种子和增量都是 1，将

authorSex 列的默认值设置为“女”,将 authorCountry 列的默认值设置为“不详”,且 authorId 和 authorName 列不允许有空值。

(5) 向 bookmanage 数据库中添加 Book 表,该表包含 bookNo、bookName、bookAuthorId、bookTypeId、bookOldPrice、bookNewPrice、bookPublish、bookInventory 以及 bookIntro 多个列。代码如下:

```
CREATE TABLE Book
(
    bookNo nvarchar(20) PRIMARY KEY NOT NULL,
    bookName nvarchar(50) NOT NULL,
    bookAuthorId int,
    bookTypeId int,
    bookOldPrice decimal NOT NULL,
    bookNewPrice decimal NOT NULL,
    bookPublish nvarchar(50),
    bookInventory int DEFAULT(0),
    bookIntro text
)
```

(6) 通过 ALTER TABLE 语句为 Book 表添加外键约束。在 Book 表中,bookAuthorId 列对应 BookAuthor 表中的 authorId 列,bookTypeId 列对应 BookType 表中的 typeId 列。代码如下:

```
ALTER TABLE Book ADD CONSTRAINT FK_Book_BookAuthor FOREIGN KEY
(bookAuthorId) REFERENCES BookAuthor(authorId);
ALTER TABLE Book ADD CONSTRAINT FK_Book_BookType FOREIGN KEY(bookTypeId)
REFERENCES BookType(typeId);
```

(7) 执行前面步骤中的语句查看效果,如图 3-19 所示。

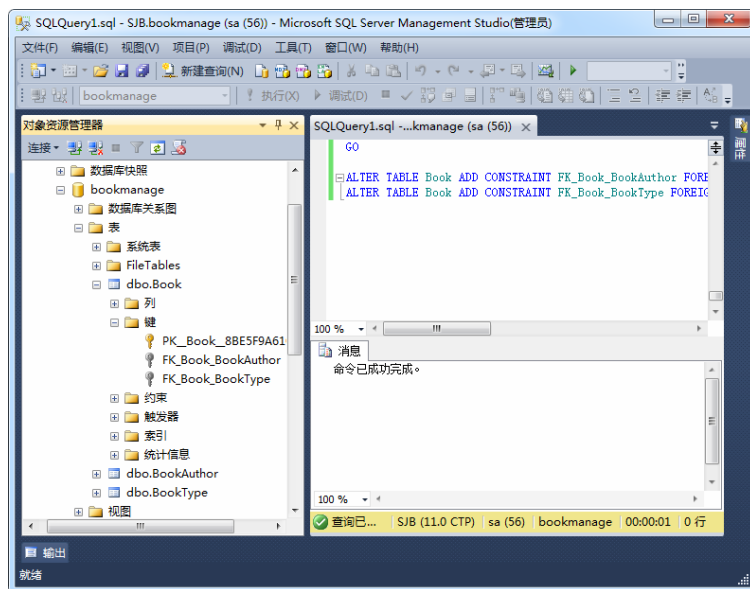


图 3-19 bookmanage 数据库

3.7 思考与练习

一、填空题

1. 用户通过 Transact-SQL 语句查看表的属性时, 可以使用_____目录视图。
2. _____和 smallmoney 是 SQL Server 中代表货币或货币值的数据类型。
3. char、varchar 和_____是 SQL Server 2012 提供的字符串类型。
4. Transact-SQL 语句通过_____创建主键约束。

二、选择题

1. 创建表的 Transact-SQL 语句是_____。
 - A. CREATE DATABASE
 - B. CREATE TABLE
 - C. ALTER DATABASE
 - D. ALTER TABLE
2. 关于 DROP TABLE 语句, 下列说法正确的是_____。
 - A. 使用 DROP TABLE 语句可以删除被 FOREIGN KEY 约束引用的表
 - B. 使用 DROP TABLE 语句只会删除表, 不会删除表中的存储数据、索引器和视图等内容
 - C. DROP TABLE 语句删除被 FOREIGN KEY 约束引用的表时, 必须先删除引用 FOREIGN KEY 约束或引用表
 - D. B 和 C 都正确
3. Unicode 字符串类型不包括_____。
 - A. nchar
 - B. nvarchar
 - C. ntext
 - D. nvarbinary
4. 以下优先级最高的数据类型是_____。
 - A. text
 - B. varchar
 - C. int
 - D. 用户自定义类型
5. 关于 SQL Server 中的约束类型, 下列说法错误的是_____。
 - A. 一个表中可以定义多个 UNIQUE 约束, 但是只能定义一个 PRIMARY KEY 约束
 - B. 一个表可以定义多个 PRIMARY KEY 约束, 但是列最多只能定义一个默认值
 - C. IDENTITY 列的数据类型只能为 tinyint、smallint、int、bigint、numeric 和 decimal。当为 numeric 和 decimal 类型时, 不允许有小数位
 - D. 一列最多有一个默认值, 其中包括 NULL 值
6. 假设某个数据库中存在 person 表和 company 表, person 表中的 inCompanyId 列与 company 表中的主键列 companyId 相对应。为 person 表中的 inCompanyId 列添加外键约束, 下列代码错误的是_____。
 - A.


```
CREATE TABLE SalesOrder(
    personId int PRIMARY KEY NOT NULL,
    personName int NOT NULL,
    inCompanyId int,
    ALTER TABLE person ADD CONSTRAINT FK_person_company FOREIGN KEY (inCompanyId) REFERENCES company (companyId)
)
```
 - B.


```
CREATE TABLE person(
    personId int PRIMARY KEY NOT NULL,
    personName int NOT NULL,
    inCompanyId int,
)
ALTER TABLE person ADD CONSTRAINT FK_person_company FOREIGN KEY (inCompanyId) REFERENCES company (companyId)
```
 - C.


```
CREATE TABLE SalesOrder(
```

```
personId int PRIMARY KEY NOT  
NULL,  
personName int NOT NULL,  
inCompanyId int,  
CONSTRAINT FK_person_company  
FOREIGN KEY (inCompanyId)  
REFERENCES company(companyId)  
)
```

D.

```
CREATE TABLE person(  
    personId int PRIMARY KEY NOT  
    NULL,  
    personName int NOT NULL,
```

```
inCompanyId int,  
FOREIGN KEY (inCompanyId)  
REFERENCES company(companyId)  
)
```

三、简答题

1. 在数据库中创建表和删除表需要使用的 Transact-SQL 语句是什么？
2. 指定列的数据类型有哪些？对这些数据类型进行说明。
3. 本章介绍的约束类型有哪些？对这些约束类型进行举例说明。