第7章 管理数据库表

主要内容

返回总目录

- 创建数据库表
- 创建和使用约束
- 自定义数据类型
- 修　改　表
- 查　看　表
- 删　除　表

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部分。管理好表也就管理好了数据库。本章将介绍如何创建和管理数据库表。

7.1 创建数据库表

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。

7.1.1 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：NULL 或 NOT NULL 属性和 IDENTITY 属性。

1．NULL 或 NOT NULL

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。在 SQL Server 中，列的缺省属性是 NOT NULL。要设置缺省属性为 NULL 或 NOT NULL，可以在 Enterprise Manager 中修改数据库属性选项中的“ANSI null default”为真或假。也可以使用如下两种语句来设定：

- set ansi_null_dflt_on 或 set ansi_null_dflt_off;
- sp_dboption database_name, ‘ANSI null default’, true/false。

2．IDENTITY

IDENTITY 属性可以使表的列包含系统自动生成的数字。这种数字在表中可以惟一标识表的每一行，即表中的每一行数据在指定为 IDENTITY 属性的列上的数字均不相同。指定了 IDENTITY 属性的列称为 IDENTITY 列。当用 IDENTITY 属性定义一个列时，可以指定一个初始值和一个增量。插入数据到含有 IDENTITY 列的表中时，初始值在插入第一行数据时使用，以后就由 SQL Server 根据上一次使用的 IDENTITY 值加上增量得到新的 IDENTITY 值。如果不指定初始值和增量值，则其缺省值均为 1。

IDENTITY 属性适用于 INT、SMALLINT、TINYINT、DECIMAL（P,0）、N：：：：：？UMERIC（P,0）数据类型的列。

> ！ 一个列不能同时具有 NULL 属性和 IDENTITY 属性，只能二者选其一。

7.1.2 用 CREATE TABLE 命令创建表

用 CREATE TABLE 命令创建表快捷、明了。其语法如下：

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

({<column_definition> | column_name AS computed_column_expression | <table_constraint>} [,...n])

[ON {filegroup | DEFAULT}]
[TEXTIMAGE_ON {filegroup | DEFAULT}]

<column_definition> ::= { column_name data_type }
[[DEFAULT constant_expression]
| [IDENTITY [(seed, increment) [NOT FOR REPLICATION]]]]
[ROWGUIDCOL]
[COLLATE < collation_name >]
[<column_constraint>] [...n]

各参数说明如下：

- database_name

指定新建的表属于哪个数据库。如果不指定数据库名，就会将所创建的表存放在当前数据库中。

- owner

指定数据库所有者的用户名。

- table_name

指定新建的表的名称，最长不超过 128 个字符。

对数据库来说，database_name.owner_name.object_name 应该是惟一的。

- column_name

指定列的名称。

- computed_column_expression

指定计算列（Computed column）的列值的表达式。表达式可以是列名、常量、变量、函数等或它们的组合。所谓计算列是一个虚拟的列，它的值并不实际存储在表中，而是通过对同一个表中其它列进行某种计算而得到的结果。例如：员工信息表中存储了员工的雇佣日期，那么员工的工龄就可以通过表达式“雇佣日期－当前日期”计算出来，则工龄列就可作为一个计算列。

- ON {filegroup | DEFAULT}

指定存储表的文件组名。如果使用了 DEFAULT 选项或省略了 ON 子句，则新建的表会存储在默认文件组中。

- TEXTIMAGE_ON

指定 TEXT、NTEXT、和 IMAGE 列的数据存储的文件组。如果无此子句，这些类型的数据就和表一起存储在相同的文件组中。

- data_type

指定列的数据类型。

- DEFAULT

指定列的缺省值。当输入数据时，如果用户没有指定列值，系统就会用设定的缺省值作为列值。如果该列没有指定缺省值但允许 NULL 值，则 NULL 值就会作为缺省值。其中缺省值可以为常数、NULL 值、SQL Server 内部函数（如 GETDATE（）函数）、NILADIC 函数等。

- constant_expression

列缺省值的常量表达式，可以为一个常量或系统函数或 NULL。

- IDENTITY

指定列为 IDENTITY 列。一个表中只能有一个 IDENTITY 列。

- seed

指定 IDENTITY 列的初始值。

- increment

指定 IDENTITY 列的增量。

- NOT FOR REPLICATION

指定列的 IDENTITY 属性在把从其它表中复制的数据插入到表中时不发生作用，即不足的生成列值，使得复制的数据行保持原来的列值。

- ROWGUIDCOL

指定列为全球惟一鉴别行号列（ROWGUIDCOL 是 Row Global Unique Identifier Column 的缩写）。此列的数据类型必须为 UNIQUEIDENTIFIER 类型。一个表中数据类型为 UNIQUEIDENTIFIER 的列中只能有一个列被定义为 ROWGUIDCOL 列。ROWGUIDCOL 属性不会使列值具有惟一性，也不会自动生成一个新的数值给插入的行。需要在 INSERT 语句中使用 NEWID（）函数或指定列的缺省值为 NEWID（）函数。

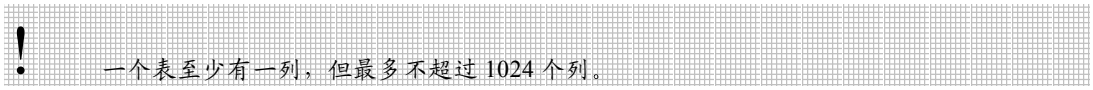
- COLLATE

指明表使用的校验方式。

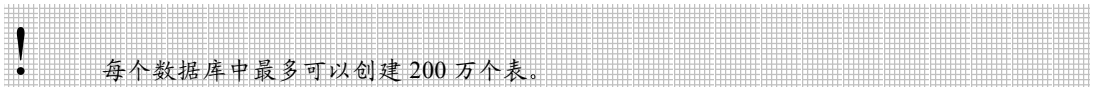
- column_constraint 和 table_constraint

指定列约束和表约束，我们将在下一节中介绍其具体定义。

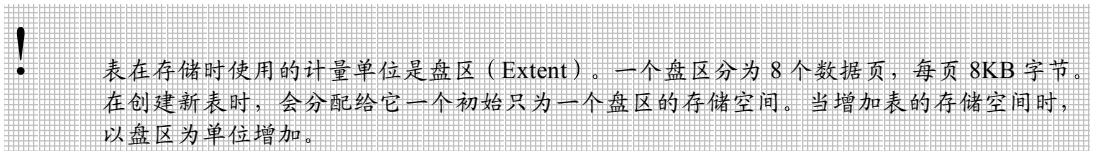
其余参数将在后面的章节中逐步讲述。



一个表至少有一列，但最多不超过 1024 个列。



每个数据库中最多可以创建 200 万个表。



表在存储时使用的计量单位是盘区（Extent）。一个盘区分为 8 个数据页，每页 8KB 字节。在创建新表时，会分配给它一个初始只为一个盘区的存储空间。当增加表的存储空间时，以盘区为单位增加。

例 7-1：创建一个商品信息表。

```
create table mydb.dbo.products3 (
    p_id    smallint identity (1000, 1) ,        /* 商品序列号自动增长 */
    p_name    char(10) not null ,
    price    money default 0.01 ,                /* 商品单价缺省值为 0.01 元 */
    quantity    smallint null ,
     sumvalue    as price*quantity               /* 商品总价值=单价*数量 */
) on [primary]
```

7.1.3 用 Enterprise Manager 创建表

在 Enterprise Manager 中创建表按以下步骤进行：

（1）在要创建表的数据库中选择“Tables”对象后，单击右键，从快捷菜单中选择“New Table”选项，或在工具栏中选择图标，即会出现如图 7-1 所示的定义列对话框。在此可以是设定表的列名、数据类型、精度、缺省值等属性。

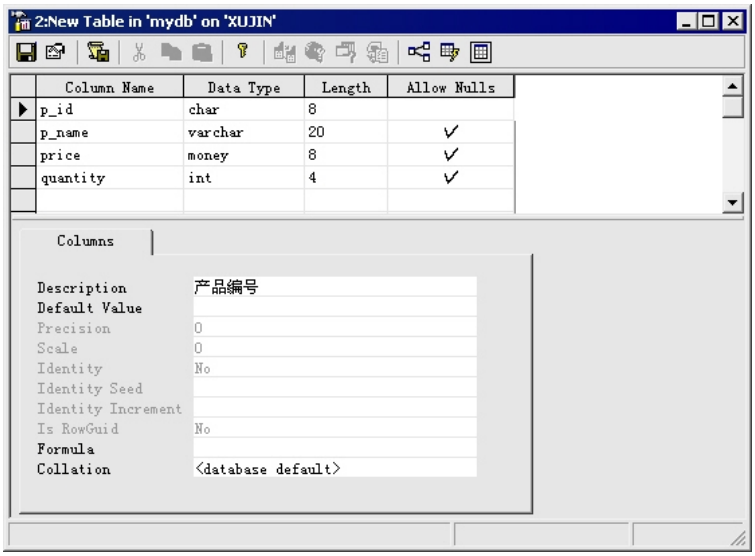


图7-1 定义列对话框

（2）单击图 7-1 工具栏中的保存按钮，即出现如图 7-2 所示的输入新建表名的对话框。

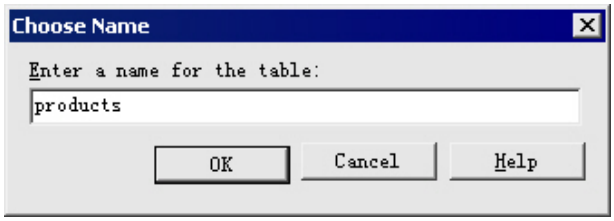


图7-2 输入新建表名对话框

（3）输入表名后，单击“OK”按钮，即会将表保存到数据库中。然后图 7-1 中工具栏右边原来不可用的几个按钮将变为可用，可以使用它们来设置表的其它信息，在以后的章节中将讲到它们的使用。

7.1.4 创建临时表

可以用 CREATE TABLE 命令创建表局部的或全局作用的临时表。其语法与创建一般表基本相同，只是在局部临时表的表名前要使用符号“#”，全局临时表的表名前要使用符号“##”，以便与一般的表相区别。由于 SQL Server 将临时表的表名存储到 Tempdb 数据库中 sysobjects 表中时，会自动在其后面添加一个系统产生的 12 位的数字后缀，因此临时表的表名最长只能指定 116 个字符，以不超过 128 个字符的命名限制。

例 7-2：创建一个局部临时表 test123。

```
create table #test123 (
    test_id    smallint ,
    test_name    char(10) ,
) on [primary]
```

7.2 创建和使用约束

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的限制条件（有关数据完整性的介绍请参见第 9 章）。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）、惟一性约束（Unique Constraint）、检查约束（Check Constraint）和缺省约束（Default Constraint）。

7.2.1 主关键字约束

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。每个表中只能有一列被指定为主关键字，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。

定义主关键字约束的语法如下：

CONSTRAINT constraint_name
PRIMARY KEY [CLUSTERED | NONCLUSTERED]
（column_name1[, column_name2,…,column_name16] ）

各参数说明如下：

- constraint_name

指定约束的名称。约束的名称在数据库中应是惟一的。如果不指定，则系统会自动生成一个约束名。

- CLUSTERED | NONCLUSTERED

指定索引类别，CLUSTERED 为缺省值。其具体信息请参见下一章。

- column_name

指定组成主关键字的列名。主关键字最多由 16 个列组成。

例 7-3：创建一个产品信息表，以产品编号和名称为主关键字

```
create table products (
        p_id      char(8) not null,
        p_name    char(10) not null ,
        price     money default 0.01 ,
        quantity    smallint null ,
       constraint pk_p_id primary key (p_id, p_name)
) on [primary]
```

7.2.2 外关键字约束

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。

定义外关键字约束的语法如下：

CONSTRAINT constraint_name
FOREIGN KEY（column_name1[, column_name2,…,column_name16] ）
REFERENCES ref_table [（ref_column1[,ref_column2,…, ref_column16] ）]
[ON DELETE { CASCADE | NO ACTION }]
[ON UPDATE { CASCADE | NO ACTION }]]
[NOT FOR REPLICATION]

各参数说明如下：

- REFERENCES

指定要建立关联的表的信息。

- ref_table

指定要建立关联的表的名称。

- ref_column

指定要建立关联的表中的相关列的名称。

- ON DELETE {CASCADE | NO ACTION}

指定在删除表中数据时，对关联表所做的相关操作。在子表中有数据行与父表中的对应数据行相关联的情况下，如果指定了值 CASCADE，则在删除父表数据行时会将子表中对应的数据行删除；如果指定的是 NO ACTION，则 SQL Server 会产生一个错误，并将父表中的删除操作回滚。NO ACTION 是缺省值。

- ON UPDATE {CASCADE | NO ACTION}

指定在更新表中数据时，对关联表所做的相关操作。在子表中有数据行与父表中的对应数据行相关联的情况下，如果指定了值 CASCADE，则在更新父表数据行时会将子表中对应的数据行更新；如果指定的是 NO ACTION，则 SQL Server 会产生一个错误，并将父

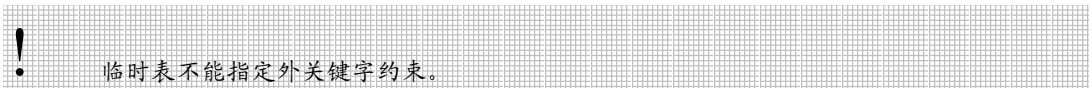
表中的更新操作回滚。NO ACTION 是缺省值。

- NOT FOR REPLICATION

指定列的外关键字约束在把从其它表中复制的数据插入到表中时不发生作用。

例 7-4：创建一个订货表，与前面创建的产品表相关联

```
create table orders(
        order_id    char(8),
        p_id    char(8),
        p_name    char(10) ,
        constraint pk_order_id primary key (order_id) ,
        foreign key(p_id, p_name) references products(p_id, p_name)
) on [primary]
```



7.2.3 惟一性约束

惟一性约束指定一个或多个列的组合的值具有惟一性，以防止在列中输入重复的值。惟一性约束指定的列可以有 NULL 属性。由于主关键字值是具有惟一性的，因此主关键字列不能再设定惟一性约束。惟一性约束最多由 16 个列组成。

定义惟一性约束的语法如下：

```
CONSTRAINT    constraint_name
    UNIQUE [CLUSTERED | NONCLUSTERED]
    （column_name1[, column_name2,…,column_name16] ）
```

例 7-5：定义一个员工信息表，其中员工的身份证号具有惟一性。

```
create table employees (
    emp_id    char(8),
    emp_name    char(10) ,
    emp_cardid char(18),
        constraint pk_emp_id primary key (emp_id),
        constraint uk_emp_cardid unique (emp_cardid)
) on [primary]
```

7.2.4 检查约束

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。可以对每个列设置符合检查。

定义检查约束的语法如下：

```
CONSTRAINT    constraint_name
     CHECK [NOT FOR REPLICATION]
```

(logical_expression)

各参数说明如下：

- NOT FOR REPLICATION

指定检查约束在把从其它表中复制的数据插入到表中时不发生作用。

- logical_expression

指定逻辑条件表达式，返回值为 TRUE 或 FALSE。

例 7-6：创建一个订货表，其中定货量必须不小于 10。

```
create table orders(
        order_id    char(8),
        p_id    char(8),
        p_name    char(10) ,
        quantity    smallint,
        constraint pk_order_id primary key (order_id),
        constraint chk_quantity    check (quantity>=10) ,
) on [primary]
```

! 对计算列不能作除检查约束外的任何约束。

7.2.5 缺省约束

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。有关绑定缺省约束的方法请参见“数据完整性”章节。

定义缺省约束的语法如下：

CONSTRAINT constraint_name
DEFAULT constant_expression [FOR column_name]

例 7-7：

```
constraint de_order_quantity default 100 for order_quantity
```

! 不能在创建表时定义缺省约束，只能向已经创建好的表中添加缺省约束。

7.2.6 列约束和表约束

对于数据库来说，约束又分为列约束（Column Constraint）和表约束（Table Constraint）。列约束作为列定义的一部分只作用于此列本身。表约束作为表定义的一部分，可以作用于多个列。

下面举例说明列约束与表约束的区别。

例 7-8：

```
create table products (
        p_id      char(8) ,
        p_name    char(10) ,
        price     money default 0.01 ,
        quantity  smallint  check  (quantity>=10) ,    /* 列约束 */
       constraint pk_p_id primary key (p_id, p_name)    /* 表约束 */
) on [primary]
```

7.3 自定义数据类型

除了使用系统提供的数据类型外，用户还可以根据需要用自定义的数据类型来定义表的列或声明变量。

7.3.1 用 Enterprise Manager 创建用户自定义数据类型

用 Enterprise Manager 创建用户自定义数据类型的方法是：在 Enterprise Manager 中选择要创建用户自定义类型的数据库，在数据库对象“User Defined Data Types”上单击右键，从开始菜单中选择“New User Defined Data Type”选项，就会出现如图 7-3 所示的定义用户自定义数据类型属性对话框。可以在其中指定要定义的数据类型的名称、继承的系统数据类型、是否允许 NULL 值等属性。单击“确定”按钮，则添加用户自定义数据类型对象到数据库中。

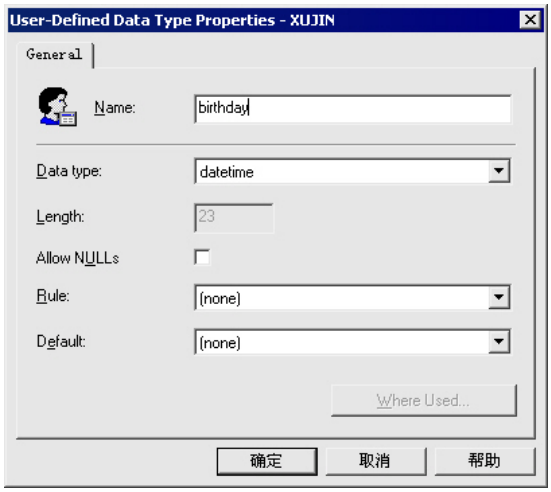


图7-3 定义用户自定义数据类型属性对话框

7.3.2 用系统存储过程 Sp_addtype 创建用户自定义数据类型

系统存储过程为用户提供了命令方式创建自定义数据类型的途径。其语法如下：

sp_addtype [@typename =] type,
[@phystype =] system_data_type
[, [@nulltype =] 'null_type']
[, [@owner =] 'owner_name']

各参数说明如下：

- [@typename =] type

指定用户定义的数据类型的名称。

- [@phystype =] system_data_type

指定相应的系统提供的数据类型的名称及定义。不能使用 TIMESTAMP 数据类型。当所使用的系统数据类型有额外说明时，需用引号将其括起来，如：‘CHAR（8）’。

- [@nulltype =] ‘null_type’

指定用户自定义的数据类型的 NULL 属性，其值可为‘NULL’、‘NOT NULL’或‘NONULL’。缺省时与系统默认的 NULL 属性相同。

- [@owner =] 'owner_name']

指定用户自定义的数据类型的所有者。

用户自定义的数据类型的名称在数据库中应是惟一的，但不同名称的用户自定义数据类型可以有相同的类型定义。在使用 CREATE TABLE 命令时，用户自定义数据类型的 NULL 属性可以被改变，但其长度定义不能更改。

例 7-9：定义生日数据类型。

```
exec sp_addtype birthday, datetime, 'not null'
```

运行结果如下：

```
(1 row(s) affected)
Type added.
```

例 7-10：定义身份证号码数据类型。

```
exec sp_addtype cardid, 'char(18)', 'not null'
```

运行结果如下：

```
(1 row(s) affected)
Type added.
```

例 7-11：定义地址数据类型。

```
exec sp_addtype address, 'varchar(100)', 'not null'
```

运行结果如下：

```
(1 row(s) affected)
Type added.
```

7.3.3 删除用户自定义数据类型

可以在 Enterprise Manager 中选择用户自定义类型后，从快捷菜单中选择“Delete”选项将其删除，也可以使用系统存储过程 Sp_droptype 将其删除。其语法如下：

sp_droptype [@typename =] ‘type’

例 7-12：

```
exec sp_droptype cardid
```

运行结果如下：

```
(1 row(s) affected)
(0 row(s) affected)
Type has been dropped.
```

! 如果删除由表或其它数据库在使用的用户自定义数据类型，将会被系统拒绝。

7.4 修改表

当表创建好后，可能根据需要要对表的列、约束等属性进行添加、删除或修改，这就需要修改表结构。

7.4.1 用 Enterprise Manager 修改

在 Enterprise Manager 中选择要进行改动的表，单击右键，从快捷菜单中选择“Design Table”选项，则会出现如图 7-4 所示的修改表结构对话框。可以在图 7-4 所示的对话框中修改列的数据类型、名称等属性或添加、删除列，也可以指定表的主关键字约束。单击工具栏中的图标  ，出现如图 7-5 所示的编辑表和约束的属性的对话框。可以在其中编辑各种约束和一些表的属性。

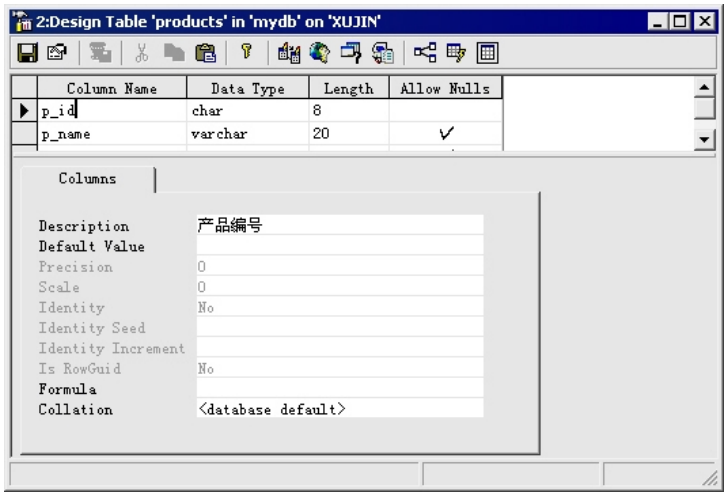


图7-4 修改表结构对话框

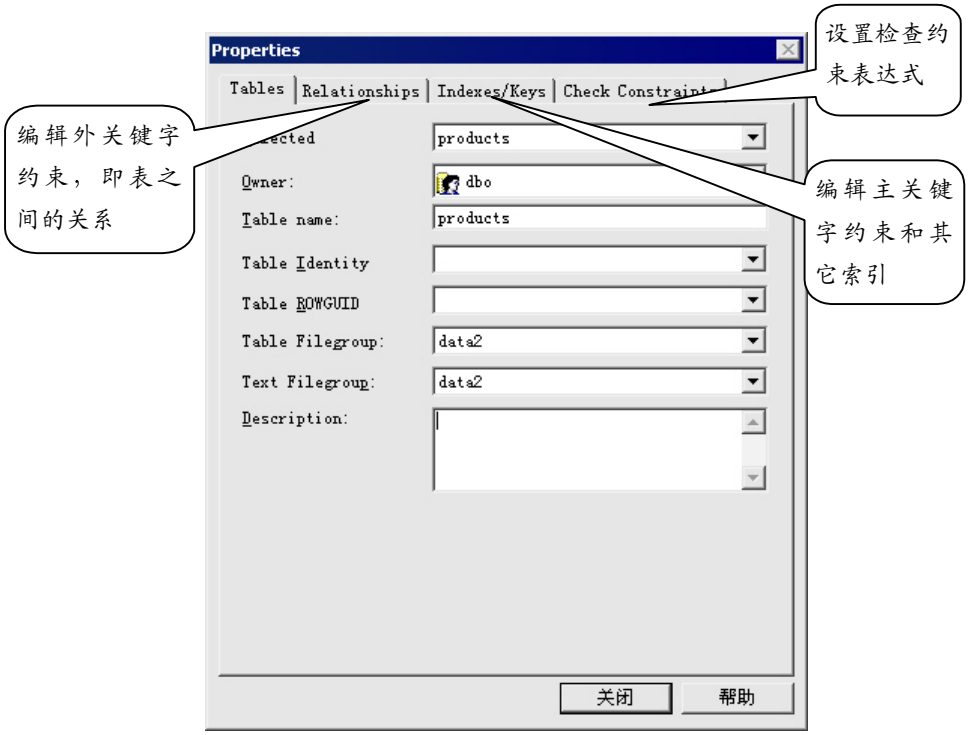


图7-5 编辑表和约束的属性对话框

7.4.2 用 ALTER TABLE 命令修改

ALTER TABLE 命令可以添加或删除表的列、约束，也可以禁用或启用已存在的约束或触发器。其语法如下：

ALTER TABLE table

```
    {    [ALTER COLUMN column_name
            {    new_data_type [ (precision[, scale] ) ]
                    [ COLLATE < collation_name > ]
                   [ NULL | NOT NULL ]
                 | {ADD | DROP} ROWGUIDCOL   } ]
       | ADD
            {    [ <column_definition> ]
              |   column_name AS computed_column_expression
            }[,...n]
       | [WITH CHECK | WITH NOCHECK] ADD
            { <table_constraint> }[,...n]
       | DROP
            {    [CONSTRAINT] constraint_name
              | COLUMN column
            }[,...n]
       | {CHECK | NOCHECK} CONSTRAINT
            {ALL | constraint_name[,...n]}
       | {ENABLE | DISABLE} TRIGGER
          {ALL | trigger_name[,...n]}
    }

<column_definition> ::= { column_name data_type }
      [ [ DEFAULT constant_expression ]
        | [ IDENTITY [(seed, increment ) [NOT FOR REPLICATION] ] ]
      ]
       [ ROWGUIDCOL ]
        [ COLLATE < collation_name > ]
         [ <column_constraint>] [ ...n]

<column_constraint> ::= [CONSTRAINT constraint_name]
{ [ NULL | NOT NULL ]
    | [  { PRIMARY KEY | UNIQUE }
          [CLUSTERED | NONCLUSTERED]
          [WITH FILLFACTOR = fillfactor]
          [ON {filegroup | DEFAULT} ] ] ]
   | [   [FOREIGN KEY]
      REFERENCES ref_table [(ref_column) ]
      [ ON DELETE { CASCADE | NO ACTION } ]
      [ ON UPDATE { CASCADE | NO ACTION } ]
```

```
    [NOT FOR REPLICATION ] ]
  | CHECK [NOT FOR REPLICATION]
    (logical_expression)}

<table_constraint> ::= [CONSTRAINT constraint_name]
{ [ { PRIMARY KEY | UNIQUE }
    [ CLUSTERED | NONCLUSTERED]
    { ( column [ ASC | DESC ] [,...n] ) }
    [ WITH FILLFACTOR = fillfactor]
    [ON {filegroup | DEFAULT} ] ]
  | FOREIGN KEY
      [(column[,...n])]
      REFERENCES ref_table [(ref_column[,...n])]
      [NOT FOR REPLICATION]
      [ ON DELETE { CASCADE | NO ACTION } ]
      [ ON UPDATE { CASCADE | NO ACTION } ]
  | CHECK [NOT FOR REPLICATION]
    (search_conditions)}
```

各参数说明如下：

- table

指定要修改的表的名称。如果表不在当前数据库中或表不属于当前的用户，就必须指明其所属的数据库名称和所有者名称。

- ALTER COLUMN

说明给出的列需要被变更或修改数据类型。如下形式的列不能修改数据类型：

✦列的数据类型为 TEXT、IMAGE、NTEXT 或 TIMESTAMP 类型；

✦表的 ROWGUIDCOL 列；

✦计算列或被计算列使用的列；

✦被复制的列；

✦用于一个索引的列。除非此列为 VARCHAR 或 VARBINARY 数据类型，并且不改变其数据类型，只加大其长度；

✦用于统计信息的列；

✦用于 PRIMARY KEY 或 FOREIGN KEY REFERENCES 约束中的列；

✦用于 CHECK 或 UNIQUE 约束的列。除非此列的数据类型长度可变动；

✦与一个缺省约束相关联的列。除非不改变数据类型，而只在其限定范围内改变数据类型的长度、精度。

- new_data_type

指定新的数据类型名称，其使用标准如下：

✦列的原数据类型应可以转换为新的数据类型；

✦新的数据类型不能为 TIMESTAMP；

✦新的数据类型允许列为 NULL 值;

✦如果原来的列是 IDENTITY 列，则新的数据类型应支持 IDENTITY 特性;

✦当前的 SET ARITHABORT 设置将被视为处于 ON 状态。

- precision

指定新数据类型的位数。

- scale

指定新数据类型的小数位数。

- NULL | NOT NULL

指明列是否允许 NULL 值。如果添加列到表中时，指定它为 NOT NULL，则必须指定此列的缺省值。选择此项后，new_data_type [(precision [, scale])]选项就必须指定。即使 precision 和 scale 选项均不变，当前的数据类型也需要指出来。

- WITH CHECK | WITH NOCHECK

指定已经存在于表中的数据是否需要使用新添加的或刚启用的 FOREIGN KEY 约束或 CHECK 约束来验证。如果不指定，WITH CHECK 作为新添加约束的缺省选项，WITH NOCHECK 作为启用旧约束的缺省选项。

- {ADD | DROP} ROWGUIDCOL

添加或删除列的 ROWGUIDCOL 属性。ROWGUIDCOL 属性只能指定给一个 UNIQUEIDENTIFIER 列。

- ADD

添加一个或多个列、计算列或表约束的定义。

- computed_column_expression

计算列的计算表达式。

- DROP { [CONSTRAINT] constraint_name | COLUMN column_name }

指定要删除的约束或列的名称。处于下列情况的列不能删除：

✦用于复制的列;

✦用于索引的列;

✦用于 CHECK、FOREIGN KEY、UNIQUE 或 PRIMARY KEY 约束的列;

✦定义了缺省约束或绑定了一个缺省值对象的列;

✦绑定了规则（Rule）的列。

- { CHECK | NOCHECK} CONSTRAINT

启用或禁用 FOREIGN KEY 或 CHECK 约束。

- ALL

使用 NOCHECK 选项禁用所有的约束，或使用 CHECK 选项启用所有的约束。

- {ENABLE | DISABLE} TRIGGER

启用或禁用触发器。

- ALL

启用或禁用选项针对所有的触发器。

- trigger_name

指定触发器名称。

其它参数与创建表和约束中所讲的相同。

例 7-13：创建一个定货商信息表，然后修改简介列的数据类型。

```
create table order_firm (
      order_firm_id char (8) primary key,
      firm_name    varchar (50) not null
     firm_introduce    char(50) null
) on [primary]
alter table order_firm
alter column firm_introduce varchar(250) null
```

例 7-14：创建一个定货表，再插入一个定货商编号列。

```
create table orders(
          order_id    char(8) ,
          p_id    char(8) foreign key references products(p_id),
          order_quantity    smallint check (order_quantity>=10),
          constraint pk_order_id primary key (order_id),
) on [primary]
alter table orders
   add order_firm_id char(8) null
       constraint fk_order_firm_id foreign key references order_firm(order_firm_id)
```

例 7-15：更改上例中的检查约束，并删除一个外关键字约束。

```
alter table orders
   add constraint chk_order_quantity check (order_quantity>=100)
   drop constraint chk_order_quantity
```

7.4.3 用存储过程 Sp_rename 修改表名和列名

Sp_rename 存储过程可以修改当前数据库中用户对象的名称，如表、列、索引、存储过程等。其语法如下：

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

其中[@objtype =] 'object_type'是要改名的对象的类型，其值可以为‘COLUMN’、‘DATABASE’、‘INDEX’、‘USERDATATYPE’、‘OBJECT’。值‘OBJECT’指代了系统表 sysobjects 中的所有对象，如表、视图、存储过程、触发器、规则、约束等。‘OBJECT’值为默认值。

例 7-16：更改 orders 表的列 p_id 名称为 products_id。

```
exec sp_rename 'orders.[p_id]', 'product_id', 'column'
```

运行结果如下：

Caution: Changing any part of an object name could break scripts and stored procedures.

The column was renamed to 'product_id'.

例 7-17：更改 orders 表的名称为 p_orders。

```
exec sp_rename 'orders', 'p_orders''
```

运行结果如下：

Caution: Changing any part of an object name could break scripts and stored procedures.

The object was renamed to 'p_orders'.

7.5 查看表

7.5.1 查看表的属性

在 Enterprise Manager 中，用右键单击要查看属性的表，从快捷菜单中选择“属性（Properties）”选项，则会出现如图 7-6 所示的表的属性对话框，从中可以看到表的大部分属性信息。应注意的是，此属性对话框与图 7-5 所示的不同，它的内容要少一些，且不能修改。点击“Permissions”按钮，还可以查看和修改表的权限。有关权限的设置请参见“安全性与用户管理”章节。

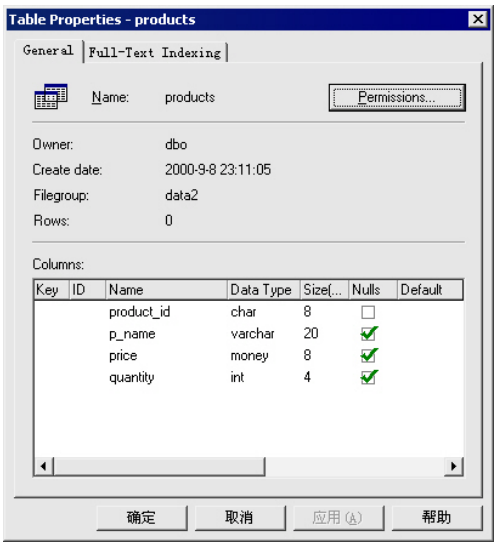
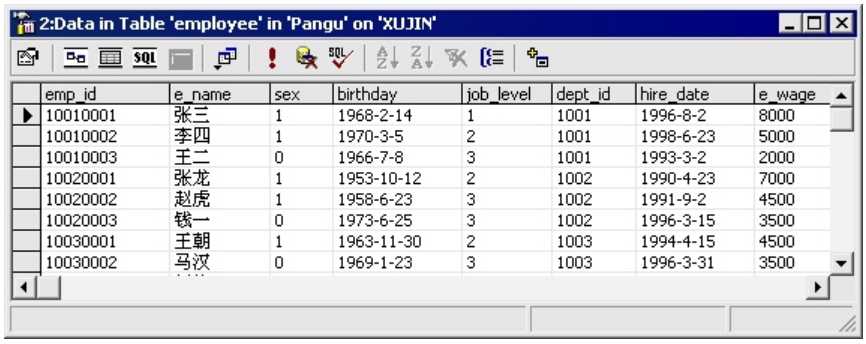


图7-6 表的属性对话框

7.5.2 查看表中的数据

在 Enterprise Manager 中用右键单击要查看属性的表，从快捷菜单中选择“Open Table”

子菜单中的“Return all rows”，则会显示表中的所有数据，出现如图 7-7 所示的对话框。



emp_id	e_name	sex	birthday	job_level	dept_id	hire_date	e_wage
10010001	张三	1	1968-2-14	1	1001	1996-8-2	8000
10010002	李四	1	1970-3-5	2	1001	1998-6-23	5000
10010003	王二	0	1966-7-8	3	1001	1993-3-2	2000
10020001	张龙	1	1953-10-12	2	1002	1990-4-23	7000
10020002	赵虎	1	1958-6-23	3	1002	1991-9-2	4500
10020003	钱一	0	1973-6-25	3	1002	1996-3-15	3500
10030001	王朝	1	1963-11-30	2	1003	1994-4-15	4500
10030002	马汉	0	1969-1-23	3	1003	1996-3-31	3500

图7-7 表中的数据

如果从“Open Table”的子菜单中选择“Return Top…”选项，则会出现如图 7-8 所示的对话框，输入一个数值，表示从表的第一行起要查看的数据行的行数，然后就会按要求返回表的数据到图 7-7 所示的对话框中。可以在对话框中修改表中的数据或添加、删除数据行。

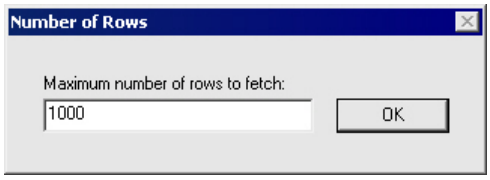


图7-8 输入要查看的数据的行数

7.5.3 用系统存储过程 Sp_help 查看表的信息

Sp_help 存储过程可以提供指定的数据库对象的信息和系统或用户定义的数据类型的信息。其语法如下：

sp_help [[@objname =] name]

Sp_help 存储过程只用于当前的数据库，其中 objname =] name 子句指定对象的名称。如果不指定对象名称，Sp_help 存储过程就会列出当前数据库中的所有对象名称、对象的所有者和对象的类型。但触发器的信息需要用 Sp_helptrigger 存储过程来显示。

例 7-18：显示当前数据库中所有对象的信息

```
exec sp_help
```

运行结果如下：

```
Name                                Owner                Object_type
-----------------------------------------------------------------------------------------------------------------
CHECK_CONSTRAINTS     INFORMATION_SCHEMA          view
…                                   …                    …
orders                              dbo                  user table
```

…　　…　　…

syscolumns　　dbo　　system table

…　　…　　…

dt_checkinobject　　dbo　　stored procedure

…　　…　　…

pk_order_id　　dbo　　primary key cns

…　　…　　…

fk_order_firm_id　　dbo　　foreign key cns

…　　…　　…

df_date　　dbo　　default (maybe cns)

…　　…　　…

CK__orders__order_qu__35BCFE0A　　dbo　　check cns

…　　…　　…

User_type　Storage_type　Length　Prec　Scale　Nullable　Default_name　Rule_name　Collation

---

address　　varchar　　100　　100　NULL　　yes　　none　　none　　Chinese_PRC_CI_AS

…　　…　　…

例 7-19：显示表 orders 的信息。

```
exec sp_help orders
```

运行结果如下：

Name　　Owner　　Type　　Created_datetime

---

orders　　dbo　　user table　　2000-08-04 15:38:10.953

Column_name　Type　Computed　Length　Prec　Scale　Nullable　TrimTrailingBlanks ……

---

order_id　　char　　no　　8　　no　　no　　……

p_id　　char　　no　　8　　yes　　no　　……

order_quantity　　smallint　　no　　2　　5　　0　　yes　　(n/a)　　……

order_firm_id　　char　　no　　8　　yes　　no　　……

7.5.4 用系统存储过程查看表的约束

1．用 Sp_helpconstraint 存储过程查看约束

Sp_helpconstraint 存储过程提供了查看表的约束信息的途径。其语法如下：

Syntax

sp_helpconstraint [@objname =] 'table'

[,[@nomsg =] 'no_message']

其中[@objname =] 'table'子句指定约束所在的表的名称；[@nomsg =] 'no_message'子句是可选项，用于打印表的名称。no_message 的数据类型为 VARCHAR(5)。

例 7-20：显示表 orders 的约束信息。

```
exec sp_helpconstraint orders
```

运行结果如下：

```
Object Name
---------------------------
orders

constraint_type              constraint_name   status_enabled   status_for_replication   constraint_keys
-------------------------------------------------------------------------------------------------------------------------
CHECK on column order_quantity   orders__qu   Enabled   Is_For_Replication   ([order_quantity] >= 10)
DEFAULT on column order_quantity   de_order_quantity   (n/a)   (n/a)   (100)
FOREIGN KEY   FK__orders__p_id__34C8D9D1   Disabled   Is_For_Replication   p_id
                                                    REFERENCES temp.dbo.products (p_id)
PRIMARY KEY (clustered)   pk_order_id   (n/a)   (n/a)   order_id
No foreign keys reference this table.
```

2．用 Sp_pkeys 存储过程查看主关键字约束

Sp_pkeys 存储过程返回当前数据库中指定表的主关键字，其语法如下：

sp_pkeys [@table_name =] 'name'

例 7-21：显示表 orders 的主关键字约束信息。

```
use pangu
exec sp_pkeys orders
```

运行结果如下：

```
TABLE_QUALIFIER   TABLE_OWNER   TABLE_NAME   COLUMN_NAME   KEY_SEQ   PK_NAME
```

--

Pangu dbo orders order_id 1 pk_order_id

(1 row(s) affected)

3．用 Sp_fkeys 存储过程查看外关键字约束

Sp_fkeys 存储过程返回当前数据库中指定表的主关键字，其语法如下：

sp_fkeys [@table_name =] 'name'

例 7-22：显示表 products 的外关键字约束信息。

```
exec sp_fkeys products
```

运行结果如下：

PKTABLE_QUALIFIER PKTABLE_NAME PKCOLUMN_NAME … FKTABLE_NAME FKCOLUMN_NAME FK_NAME … PK_NAME …

---

temp products p_id … orders p_id … PK_products …

7.6 删除表

7.6.1 用 Enterprise Manager 删除

在 Enterprise Manager 中用右键单击要删除的表，从快捷菜单中选择“删除（Delete）”选项，则会出现如图 7-9 所示的删除对象对话框。单击“Drop All”按钮，即可以删除表。单击“Show Dependencies”按钮，即会出现如图 7-10 所示的对话框。它列出了表所依靠的对象和依赖于表的对象。当有对象依赖于表时，就不能删除表。

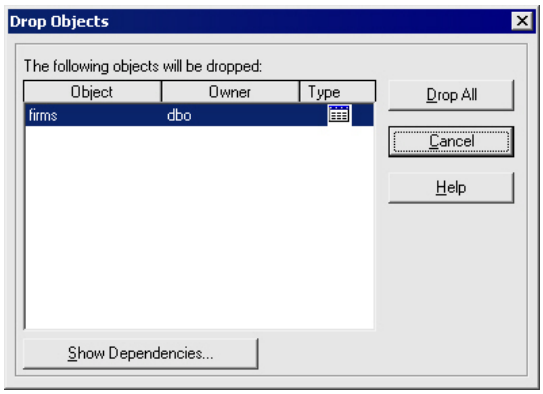


图7-9 删除对象对话框

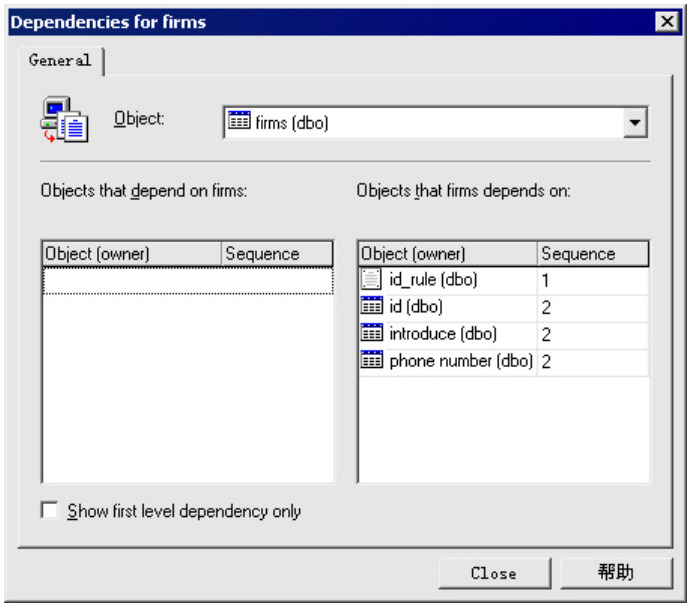


图7-10 与表相关联的对象

7.6.2 用 DROP TABLE 命令删除

DROP TABLE 命令可以删除一个表和表中的数据及其与表有关的所有索引、触发器、约束、许可对象（与表相关的视图和存储过程需要用 DROP VIEW 和 DROP PROCEDURE 命令来删除）。

DROP TABLE 命令的语法如下：

DROP TABLE table_name

要删除的表如果不在当前数据库中，则应在 table_name 中指明其所属数据库和用户名。在删除一个表之前要先删除与此表相关联的表中的外关键字约束。当删除表后，绑定的规则或缺省值会自动松绑。

> ! 不能删除系统表。

例 7-23：删除 mydb 数据库中的表 orders1。

```
drop table mydb.dbo.orders1
```

7.7 本章小结

本章介绍了创建和管理数据库表的相关知识，有关更改表中数据的详细介绍请参见第 11 章“数据库更新”。