第 5 课
数据表完整性约束

创建数据库和表之后便可以向表中存储数据，但是由于数据是从外界输入的，而数据的输入会发生输入无效或错误信息。为了保证输入的数据符合规定，SQL Server 2008 提供了大量的完整性约束。这些约束应用于基表，基表使用约束确保表中值的正确性。本课将详细介绍 SQL Server 2008 中应用于基表的各种列约束，以及默认值和规则的应用。

本课学习目标：

- ❑ 了解维护数据完整性的方法
- ❑ 理解空和非空的概念及约束方法
- ❑ 掌握自动编号约束里起始值和增量的设置
- ❑ 掌握对列应用主键和外键约束的方法
- ❑ 熟悉惟一性约束、验证约束和默认值约束的使用
- ❑ 掌握默认值对象的创建、绑定以及删除操作
- ❑ 掌握规则对象的创建、绑定以及删除操作

5.1 数据完整性概述

数据完整性是指存储在数据库中的所有数据值均正确的状态。如果数据库中存储有不正确的数据值，则该数据库称为已丧失数据完整性。

5.1.1 数据完整性简介

广义上来说，数据完整性是指数据库中数据的准确性和一致性。数据完整性是衡量数据库中数据质量好坏的一种标志，是确保数据库中数据一致、正确以及符合企业规则的一种思想。它可以使无序的数据条理化，确保正确的数据被存放在正确位置的一种手段。

满足完整性要求的数据必须具有以下三个特点。

1．数据的值正确无误

首先数据类型必须正确，其次数据的值必须处于正确的范围内。例如，在“图书管理系统”数据库的“图书明细表”中，“出版日期”一列必须满足取值范围在当前日期之前。

2．数据的存在必须确保同一表格数据之间的和谐关系

例如，在“图书明细表”的“图书编号”一列中每一个编号对应一本图书不可能将其编号对应多本图书。

3．数据的存在必须能确保维护不同表之间的和谐关系

例如，在“图书明细表”中“作者编号”一列对应“作者表”中的“作者编号”一列。在“图书明细表”中“作者编号”列所对应“作者表”中的作者编号及相关信息。

5.1.2 数据完整性分类

数据完整性是指数据的精确性和可靠性。它是为防止数据库中存在不符合语义规定的数据和防止因错误信息地输入输出造成无效操作或错误信息而提出的。数据完整性分为四类：实体完整性（Entity Integrity）、域完整性（Domain Integrity）、参照完整性（Referential Integrity）、用户定义的完整性（User defined Integrity）。

1．实体完整性

实体完整性规定表的每一行在表中是惟一的实体。实体就是数据库所要表示的一个实际的物体或事件。实体完整性要求主键的组件不能为空值。即单列主键不接受空值，复合主键的任何列也不能接受空值。

实体完整性约束来源于关系模型，而不是来源于任何特殊的应用程序的要求。实体完整性不同于其他数据库管理模型中域约束的方式，这种对于主键不能包含空值要求的原因即真实的实体通过用于惟一标识符的主键相互区分。

实体完整性的完整性问题在于设计问题，用户在设计数据库时，应该通过指定一个主键来保证实体的完整性，该主键在设计数据库时不能接受空值。例如，在“图书管理系统”数据库的“图书明细表”中，是以“图书编号”列为主键来约束其完整性。

2．域完整性

域完整性是指数据库表中的列必须满足某种特定的数据类型或约束。其中约束又包括取值范围、精度等规定。表中的 CHECK、FOREIGN KEY 约束和 DEFAULT、NOT NULL 定义都属于域完整性的范畴。

3．参照完整性

参照完整性是指两个表的主键和外键的数据应对应一致。它确保了有主键的表中对应其他表的

外键的行存在，即保证了表之间数据的一致性，防止了数据丢失或无意义的数据在数据库中扩散。参照完整性是建立在外键和主键之间或外键和唯一性关键字之间的关系上的。

在 SQL Server 中，参照完整性的作用表现在如下几个方面。

- 禁止在从表中插入包含主表中不存在的关键字的数据行。
- 禁止会导致从表中的相应值孤立的主表中的外键值改变。
- 禁止删除在从表中的有对应记录的主表记录。

4．用户定义的完整性

这种类型完整性由用户根据实际应用中的需要自行定义。可以用来实现用户定义完整性的方法有规则（Rule）、触发器（Trigger）、存储过程（Stored Procedure）和数据表创建时可以使用的所有约束（Constraint）。

提示

通过使用这些强制的完整性定义、数据库管理系统将提供更加可靠的数据，同时避免在多个用户同时操作数据库时可能发生的数据不一致。

5.2 列约束

约束是 SQL Server 2008 提供的自动保持数据库完整性的一种方法，它通过限制字段中的数据、记录中的数据和表之间的数据来保证数据的完整性。

在 SQL Server 中约束是定义在表和列上的，该约束可以被定义为列定义的一部分，或者被定义为表定义中的一个元素。

5.2.1 非空约束

所谓非空约束就是指限制一个列不允许有空值，与它对应的是空值约束，即 NULL 与 NOT NULL 约束。NULL 表示允许列为空，NOT NULL 表示不允许列为空。

列的为空性决定表中的行是否可以为该列包含空值。出现 NULL 通常是表示值未知或未定义。空值（或 NULL）与零、空白或者长度为零的字符串不同。NULL 的意思是没有输入。NOT NULL 则表示不允许为空值，即该列必须输入数据。

如果使用 NULL 约束，需要注意以下几点。

- 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。
- 用关键字 NULL 定义的列也接受用户的 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。
- NULL 值不应放在引号内，否则会被解释为字符串"NULL"而不是空值。

技巧

指定某一列不允许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。如果不允许空值，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。

【练习 1】

指定一个列是否可以为空最简单的方法是，在 SQL Server Management Studio 的【表设计器】窗口中进行设置。如图 5-1 所示为 MedicineClass 表的【表设计器】窗口。

在该窗口中表的每个列都对应一个【允许 Null 值】复选框，通过启用【允许 Null 值】复选框表示该列允许为空，否则表示不允许为空。

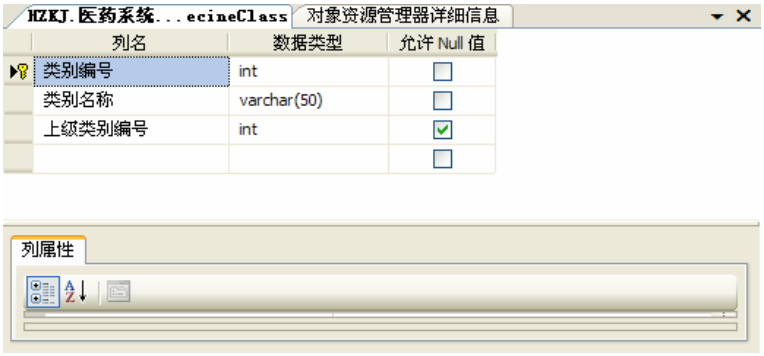

图 5-1　创建非空列

【练习 2】

在使用 CREATE TABLE 语句创建表时也可以指列的非空性。例如，要创建如图 5-1 所示的 MedicineClass 表，CREATE TABLE 实现语句如下。

```
CREATE TABLE MedicineClass
(
类别编号 int NOT NULL,
类别名称 varchar(50) NOT NULL,
上级类别编号 int NULL
)
```

如上述语句所示，通过在列的数据类型后使用 NOT NULL 关键字指定列不能为空，使用 NULL 关键字指定列允许为空。

【练习 3】

假设 MedicineClass 表已经存在，现在要将上级类别编号列修改为不允许为空，语句如下。

```
ALTER TABLE MedicineClass
ALTER
COLUMN 上级类别编号 int NOT NULL
```

注意

将 NULL 修改为 NOT NULL 时，必须保证该列数据没有空值，否则会出错。

5.2.2　自动编号约束

有些时候，一个信息表中可能会没有一个能够惟一确定这条记录的字段。例如一个用户信息表就没有办法使用用户信息的某个属性惟一确定某个用户。如果使用姓名可能会存在重名的情况；如果使用身份证号，可能存在缺少该属性的情况（比如用户忘记带身份证之类的情况）。所以通常在遇到这种情况的时候，会采取数字编号的方式，例如第一个录入的用户编号是 1，第二个录入的用户编号是 2，依次类推。在 SQL Server 2008 中可以创建自动编号的列来实现这种功能。

自动编号的列又称为标识列或 IDENTITY 约束。就像等差数列一样，依次增加一个增量。IDENTITY 约束就是为那些数值顺序递增的列准备的约束，自动完成数值添加。例如报考时用到的【考生编号】字段，就可以设置标识列，按添加顺序依次加 1。

在使用 IDENTITY 时需要注意以下几点。

- 标识数据不能由用户输入，用户只需要填写【标识种子】和【标识增量】，系统自动生成数据并填入表。

- 标识列第一条记录称为【标识种子】，依次增加的数称为【标识增量】。
- 每个表只能有一个标识列。
- 【标识种子】和【标识增量】都是非零整数，位数等于或小于 10。
- 标识列的数据类型只能是 tinyint、smallint、int、bigint、numeric、decimal。并且当数据类型为 numeric 和 decimal 时，不能有小数位。

【练习 4】

在创建表或者修改表时，通过【表设计器】窗口可以很方便地将某列设置为标识列。方法是：首先选定某列，然后在【列属性】区域将【标识规范】节点展开，第一行就是设置是否将列设为标识。单击该行，右侧出现下拉菜单标记，单击展开下拉菜单，并选择【是】选项。此时【标识规范】节点下的【标识增量】属性、【标识种子】属性和【不用于复制】属性显示为可编辑状态，直接编辑相关属性即可，如图 5-2 所示。

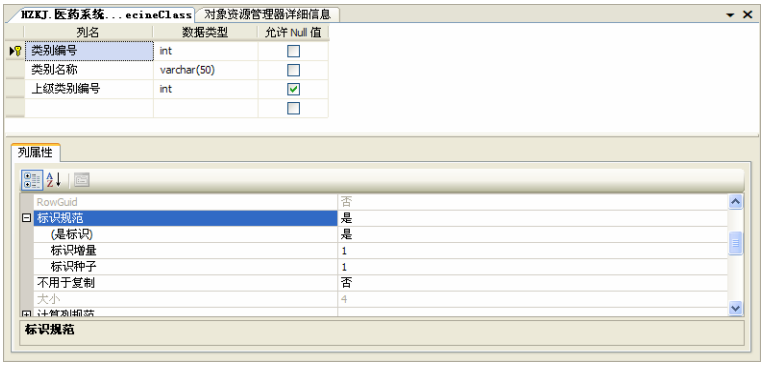

图 5-2 标识列设置

【练习 5】

使用 CREATE TABLE 指定标识符的方法是使用 IDENTITY 关键字，并同时指定标识增量和标识种子属性或者同时不指定。在不指定的情况下，默认两者均为 1。

例如，对于如图 5-2 所示的类别编号列，使用 CREATE TABLE 的实现语句如下。

```
CREATE TABLE MedicineClass
(
类别编号 int IDENTITY(1,1),
类别名称 varchar(50),
上级类别编号 int
)
```

5.2.3 主键约束

主键（PRIMARY KEY）是使用数据表中的一列或多列来惟一标识一条记录。也就是说，在一个数据表中不能存在主键完全相同的两条记录，而且位于主键中的数据必须是确定的数据，不可以为 NULL。

警告

在同一张表中，可能存在不只一个列（或组合）可以惟一地标识表中的数据。这些列（或组合）被称为候选键。数据库设计人员可以根据需求从候选键中挑选一个最为合适的列（或组合）作为表的主键。

每个表中只能有一个列（或组合）被定义为主键约束，所以该列不能包含有空值，并且 IMAGE 和 TEXT 类型的列不能定义为主键。

【练习 6】

在 SQL Server Management Studio 中管理主键的方法是在【表设计器】窗口中右击要设置为主键的列选择【设置主键】命令，如图 5-3 所示。对于已经是主键的列右击可以选择【删除主键】命令移除主键，如图 5-4 所示。

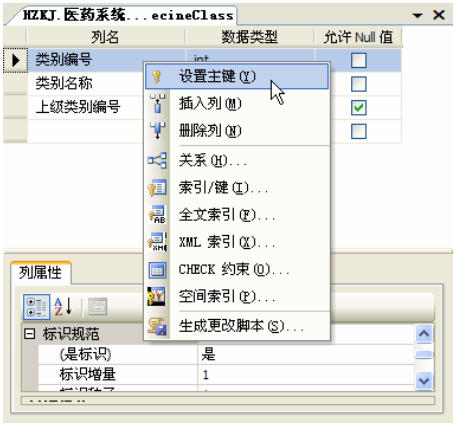

图 5-3　设置主键

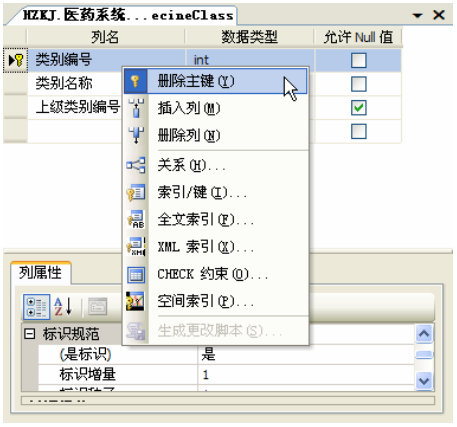

图 5-4　删除主键

注意

对于创建好的表，选择主键列时要确定不能有重复数据且没有空值，否则会出错。

【练习 7】

在使用 CREATE TABLE 创建表时，可以使用 PRIMARY KEY 关键字设置主键列。例如，下面语句在创建 MedicineClass 表时将类别编号设置为主键。

```
CREATE TABLE MedicineClass
(
类别编号 int PRIMARY KEY,
类别名称 varchar(50) ,
上级类别编号 int
)
```

【练习 8】

对于现有的表可以使用 ALTER TABLE 语句来更改列为主键，这里也要保证主键列中没有重复值和空值。

例如，下面语句将 MedicineClass 表的类别编号列设置为主键。

```
ALTER TABLE MedicineClass
ADD CONSTRAINT 类别编号 PRIMARY KEY(类别编号)
```

【练习 9】

将 MedicineClass 表中类别编号列的主键删除，语句如下。

```
ALTER TABLE MedicineClass
DROP CONSTRAINT 类别编号
```

5.2.4　外键约束

外键约束又叫 FOREIGN KEY 约束，它保证了数据库中各个表中数据的一致性和正确性。将一个表的一列（或列组合）定义为引用其他表的主键或惟一约束列，则引用表中的这个列（或列组合）

就称为外键。被引用的表称为主键约束（或惟一约束）表；引用表称为外键约束表。

1．在创建表的时候创建 FOREIGN KEY 约束

使用语法如下所示。

```
CREATE TABLE 外键表名称(
字段 数据类型 PRIMARY KEY,
字段 数据类型,
CONSTRAINT 约束名
FOREIGN KEY (外键表外键字段名)
REFERENCES 主键表名(主键表主键字段名)
)
```

【练习 10】

在 MedicineClass 表中类别编号列是主键。现在要创建 MedicineInfo 表，且要求 MedicineInfo 表中类别编号列作为外键关联 MedicineClass 表的类别编号列，使用语句如下。

```
CREATE TABLE MedicineInfo
(
药品编号 int not null,
药品名称 varchar(50) ,
类别编号 int,
CONSTRAINT 类别编号外键关联
FOREIGN KEY (类别编号)
REFERENCES MedicineClass(类别编号)
)
```

2．对现有表创建 FOREIGN KEY 约束

使用查询语句如下所示。

```
ALTER TABLE 外键表名
WITH CHECK
ADD FOREIGN KEY(外键字段名) REFERENCES 主键表名(主键)
```

【练习 11】

修改现有的 MedicineInfo 表，将类别编号列作为外键关联 MedicineClass 表的类别编号列，实现语句如下。

```
ALTER TABLE MedicineInfo
WITH CHECK
ADD FOREIGN KEY(类别编号)
REFERENCES MedicineClass(类别编号)
```

3．删除 FOREIGN KEY 约束

使用 DROP 关键字删除约束，语法如下所示。

```
ALTER TABLE 表名称
DROP
CONSTRAINT 外键约束名
```

【练习 12】

假设要删除 MedicineInfo 表中的 FOREIGN KEY 约束，语句如下。

```
ALTER TABLE MedicineInfo
DROP
CONSTRAINT 类别编号外键关联
```

5.2.5 惟一性约束

惟一性约束（UNIQUE）指定一个或多个列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。由于主键值是具有惟一性的，因此主键列不能再设定惟一性约束。

尽管 UNIQUE 约束和 PRIMARY KEY 约束都强制惟一性，但如果要强制一列或多列组合（不是主键）的惟一性时应使用 UNIQUE 约束而不是 PRIMARY KEY 约束。

UNIQUE 约束和 PRIMARY KEY 约束的区别如下所示。

- 可以对一个表定义多个 UNIQUE 约束，但只能定义一个 PRIMARY KEY 约束。
- UNIQUE 约束允许 NULL 值，这一点与 PRIMARY KEY 约束不同。不过当与参与 UNIQUE 约束的任何值一起使用时，每列只允许一个空值。
- FOREIGN KEY 约束可以引用 UNIQUE 约束。

【练习 13】

使用 SQL Server 2008 的【表设计器】窗口创建 UNIQUE 约束的步骤如下。

（1）在【表设计器】窗口上单击工具栏中的【管理索引和键】按钮，或者在选定列的右键菜单中选择【索引/键】命令打开【索引/键】对话框，如图 5-5 所示。

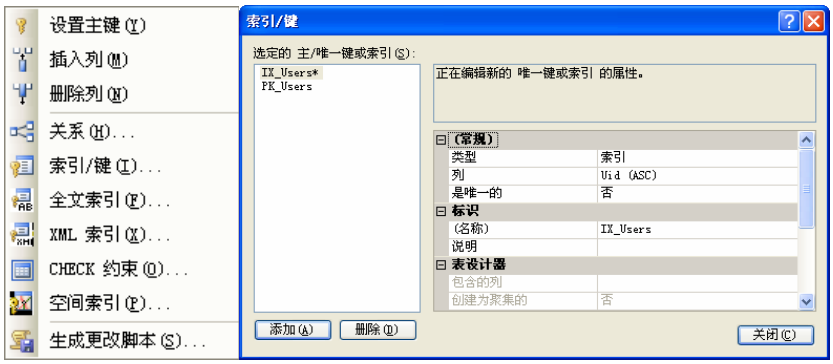

图 5-5 惟一性设置

（2）如图 5-5 所示的对话框中左边显示列表中已经存在的主键约束。单击【添加】按钮创建一个 UNIQUE 约束。

（3）在右侧编辑新建约束的属性。单击右侧列表中【列】设置项右侧的按钮，在打开的【索引列】对话框中进行设计。

（4）删除 UNIQUE 约束的方法是：同样打开【索引/键】对话框，在列表中选择要删除的约束，单击【删除】按钮删除完成关闭对话框返回，完成约束删除。

【练习 14】

在使用语句创建表时定义惟一性约束的语法如下所示。

```
CREATE TABLE 表名(
字段名 1 字段类型,
字段名 2 字段类型,
CONSTRAINT 约束名
```

```
UNIQUE(字段名 1,字段名 2)
)
```

创建一个 EmployeeInfo 表，列包含：员工编号、姓名、职称、性别、出生日期、参加工作时间、电话号码、地址和邮箱，并将姓名和电话号码设置为 UNIQUE 约束。

```
CREATE TABLE EmployeeInfo
(
员工编号 int PRIMARY KEY,
姓名 varchar(50) ,
职称 varchar(50),
性别 varchar(4) ,
出生日期 datetime,
参加工作时间 datetime ,
电话号码 varchar(50) ,
地址 varchar(50),
邮箱 varchar(50)
CONSTRAINT UNIQUE 约束
UNIQUE(姓名,电话号码)
)
```

【练习 15】

为已经存在的表设置惟一性索引，必须保证被选择设置 UNIQUE 约束的列或列的集合上没有重复值。

例如，将 EmployeeInfo 表中的邮箱字段设置为 UNIQUE 约束，查询语句如下。

```
ALTER TABLE EmployeeInfo
ADD
CONSTRAINT 邮箱约束
UNIQUE NONCLUSTERED (邮箱)
```

【练习 16】

将 EmployeeInfo 表中的邮箱约束删除，语句如下。

```
ALTER TABLE EmployeeInfo
DROP
CONSTRAINT 邮箱约束
```

5.2.6 验证约束

数据验证约束又称做 CHECK 约束，它通过给定条件（逻辑表达式）检查输入数据是否符合要求，依此来维护数据完整性。例如限制用户注册的用户名必须是字母和数字组成并以字母开头。

1．界面操作表的 CHECK 约束

在 SQL Server Management Studio 中打开【表设计器】窗口，从工具栏中单击【管理 Check 约束】按钮打开【CHECK 约束】对话框。第一次创建时这里为空，单击【添加】按钮系统自动命名并添加了一个 CHECK 约束，如图 5-6 所示。

在对话框中编辑【表达式】、【名称】等，并单击【关闭】按钮。这里的【表达式】就是一个逻辑表达式。例如，性别必须是“男”或者“女”，可以写为“性别 in ('男','女')”。

该约束添加完成后再向 EmployeeInfo 表中插入记录时将会执行该约束检查，如图 5-7 所示。

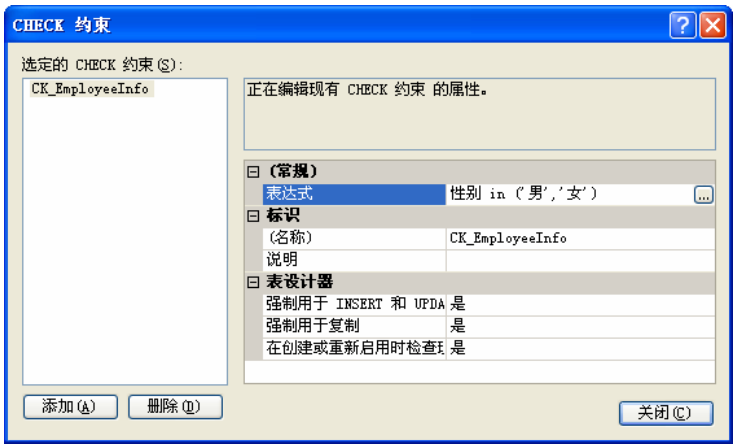

图 5-6 【CHECK 约束】对话框

图 5-7 检查数据合法性

提示

一个表或列可以存在多个 CHECK 约束，但是要保证这些验证不矛盾。

2．使用查询语句管理 CHECK 约束

创建表的时候对表定义表级别 CHECK 约束，语法如下所示。

```
CREATE TABLE 表名
(
字段 1 字段类型
CONSTRAINT 约束名
CHECK 验证表达式
)
```

CHECK 验证表达式可以有一个或多个。使用多个的时候可以用 AND 或 OR 连接，也可以用多个 CHECK 约束语句表达。

【练习 17】

创建一个用户表，并使用 CHECK 约束使年龄必须在 18～45 之间，语句如下。

```
CREATE TABLE 用户表(
用户编号 int PRIMARY KEY,
姓名 varchar(50) NOT NULL,
年龄 int NOT NULL
CONSTRAINT 检查年龄约束
CHECK (年龄>=18 AND 年龄<=45)
)
```

上述语句定义的是表级 CHECK 约束，也可以直接将 CHECK 约束写在列定之后，语句如下。

```
CREATE TABLE 用户表(
用户编号 int PRIMARY KEY,
姓名 varchar(50) NOT NULL,
年龄 int NOT NULL CHECK (年龄>=18 AND 年龄<=45)
)
```

【练习 18】

为用户表添加 CHECK 约束，使用户编号列必须大于 0。

```
ALTER TABLE 用户表
WITH CHECK
ADD
CONSTRAINT 用户编号 Check
CHECK (用户编号>0)
```

5.2.7 默认值约束

默认值约束也称为 DEFAULT 约束。将常用的数据值定义为默认值，可以节省用户输入时间，在非空的字段中定义默认值可以减少错误发生。

默认值可以像约束一样针对一个具体对象，也可以像数据库对象一样单独定义并绑定到其他对象。

在向表中插入数据时，若没有指定某一列字段的数值，则该字段的数值有以下几种情况。

- 如果该字段定义有默认值，则系统将默认值插入字段。
- 如果该字段定义没有默认值，但允许空，则插入空值。
- 如果该字段定义没有默认值，又不允许空，则报错。

如果使用 DEFAULT 约束，需要注意以下几种情况。

- DEFAULT 约束定义的默认值仅在执行 INSERT 操作插入数据时生效。
- 一列最多有一个默认值，其中包括 NULL 值。
- 具有 IDENTITY 属性或 TIMESTAMP 数据类型属性的列不能使用默认值，text 和 image 类型的列只能以 NULL 为默认值。

【练习 19】

设置默认值最简单的方法是在【表设计器】窗口中进行操作。方法是从【列属性】选项卡下展开【常规】节点，然后在【默认值或绑定】选项所在行单击，再到右边单元格编辑常量表达式，如图 5-8 所示。

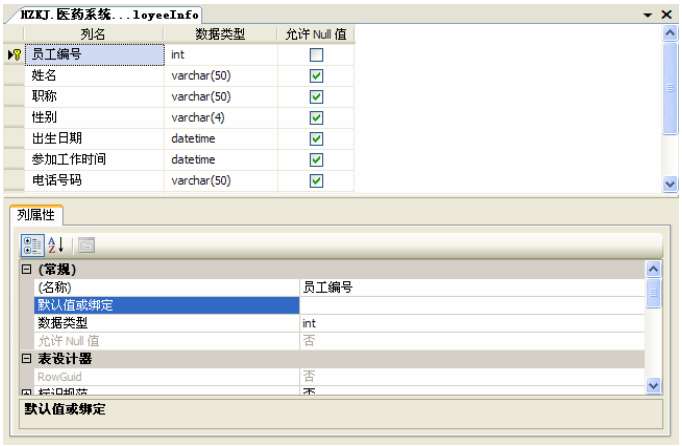

图 5-8 字段默认值

技巧

这里的常量表达式可以是具体数据值，也可以是有返回值的函数等。例如函数 GETDATE()用来返回当前时间，但是要符合该字段的数据类型及定义在该字段上的约束。

【练习 20】

创建一个会员表，使会员积分默认值为 100，使用 DEFAULT 约束的实现语句如下。

```
CREATE TABLE 会员表(
会员编号 int IDENTITY(1,1),
昵称 varchar(50) NOT NULL,
密码 varchar(50) NOT NULL,
邮箱 varchar(50) NOT NULL,
积分 int DEFAULT 100 NOT NULL
)
```

【练习 21】

为会员表添加 DEFAULT 约束，使会员密码默认为“000000”，实现语句如下。

```
ALTER TABLE 会员表
ADD
CONSTRAINT 默认密码
DEFAULT '000000' FOR 密码
```

【练习 22】

将会员表中的默认密码约束删除，实现语句如下。

```
ALTER TABLE 会员表
DROP
CONSTRAINT 默认密码
```

5.3 默认值

在 5.2.7 小节中讲解了如何在列中使用默认值约束，该约束只能应用到列上，而且对于有相同默认值要求的列需要创建多个默认值约束。为此 SQL Server 2008 提供了默认值对象，该对象一旦创建便可以重复使用。

下面详细介绍默认值对象的创建、绑定、查看及删除操作。

5.3.1 创建默认值

创建默认值对象使用的是 CREATE DEFAULT 语句，具体语法如下。

```
CREATE DEFAULT 默认值名称
AS 常量表达式
```

【练习 23】

在 medicine 数据库创建名为 Zero 的默认值，使用 0 为常量表达式，实现语句如下。

```
USE medicine
GO
CREATE DEFAULT Zero
```

```
AS 0
```

在这里要注意，默认值的定义不能包含列名，需要绑定到字段或是其他数据库对象才能使用。一个列只能绑定一个默认值，且该列最好不是惟一性列。

在【对象资源管理器】窗口中展开 medicine 数据库节点，再展开【可编辑性】|【默认值】节点，将会看到已经创建的默认值，如图 5-9 所示。

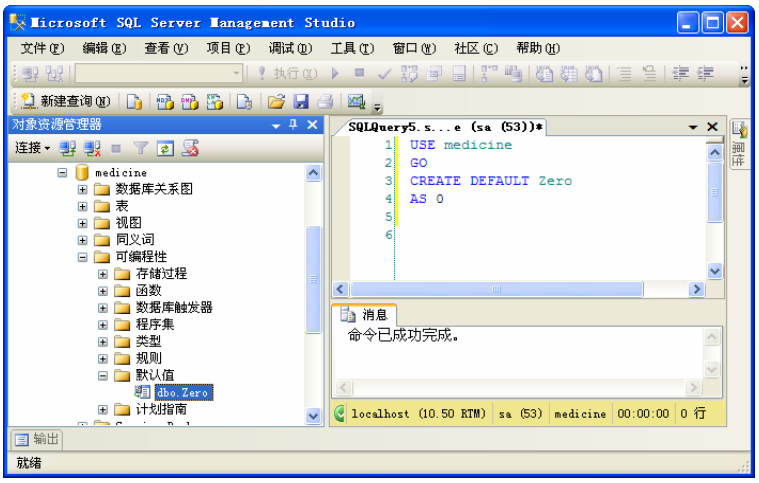

图 5-9 创建默认值

5.3.2 绑定默认值

默认值创建之后还不能立即使用，必须绑定到列才能生效。使用系统存储过程 sp_bindefault 实现默认值的绑定，具体语法如下所示。

```
sp_bindefault 默认值名称,列名.字段名
```

【练习 24】

在 medicine 数据库将 Zero 默认值绑定到 MedicineBigClass 表的 ParentId 字段，语句如下。

```
USE medicine
GO
sp_bindefault Zero,'MedicineBigClass.ParentId'
```

【练习 25】

若某列不再需要默认值，可以使用系统存储过程 sp_unbindefault 解决绑定。下面的语句解除了 MedicineBigClass 表上 ParentId 字段的默认值绑定。

```
USE medicine
GO
sp_unbindefault 'MedicineBigClass.ParentId'
```

5.3.3 查看默认值

使用【对象资源管理器】窗口展开数据库下的【可编程性】|【默认值】节点，然后在要查看的默认值名称上单击鼠标右键。从快捷菜单【编写默认值脚本为】的子菜单中可以选择的有【CREATE 到】、【DROP 到】和【DROP 和 CREATE 到】三个选项。鼠标放在它们任意一个上面选择【新查

询编辑器窗口】命令，接着在新创建的查询编辑器窗口便可以看到已经创建完成的 Zero 默认值，如图 5-10 所示。

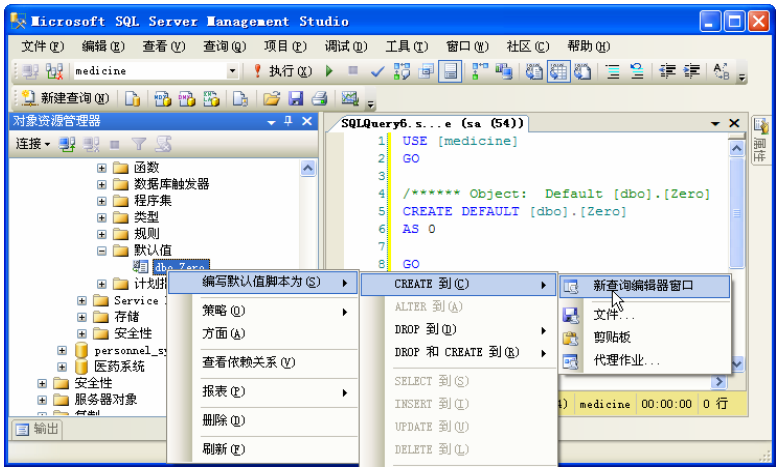

图 5-10　界面查看默认值

【练习 26】

使用 sp_help 存储过程查询 medicine 数据库 Zero 默认值，实现语句如下。

```
USE medicine
GO
sp_help Zero
```

查询结果如图 5-11 所示。

【练习 27】

使用 sp_helptext 存储过程查询 medicine 数据库 Zero 默认值，实现语句如下。

```
USE medicine
GO
sp_helptext Zero
```

查询结果如图 5-12 所示。

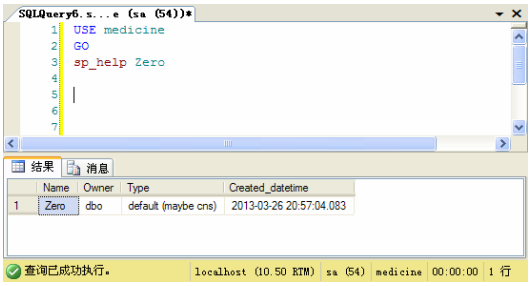

图 5-11　使用 sp_help 查看默认值

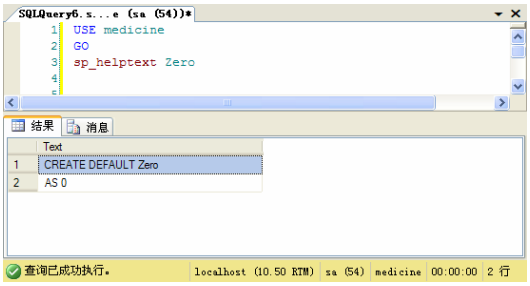

图 5-12　使用 sp_helptext 查看默认值

5.3.4　删除默认值

默认值不需要时就删除，使用 DROP DEFAULT 语句删除默认值。这里要保证默认值没有被绑定，否则该默认值尚在使用中，将无法删除。

【练习 28】

使用 DROP DEFAULT 语句删除 medicine 数据库 Zero 默认值，实现语句如下。

```
USE medicine
GO
DROP DEFAULT Zero
```

5.4 规则

规则是独立的 SQL Server 对象，它跟表和视图一样是数据库的组成部分。规则的作用和 CHECK 约束类似，用于完成对数据值的检验。它可以关联到多个表，在数据库中有数据插入和修改时，验证新数据是否符合规则，是实现域完整性的方式之一。

规则与 CHECK 约束的主要区别在于一列只能绑定一个规则，但却可以设置多个 CHECK 约束。

5.4.1 创建规则

使用 CREATE RULE 语句创建规则，语法如下所示。

```
CREATE RULE 规则名称
AS
条件表达式
```

这里的条件表达式同样使用逻辑表达式，与 CHECK 条件表达式不同的是以下内容。

- 表达式不能包含列名或其他数据库对象名。
- 表达式中要有一个以@开头的变量，代表用户的输入数据，可以看做是代替 WHERE 后面的列名。

【练习 29】

在 medicine 数据库中定义一个规则 CheckTime，限制输入的时间值必须小于当前时间，实现语句如下。

```
USE medicine
GO
CREATE RULE CheckTime
AS
@value <getdate()
```

5.4.2 绑定规则

规则和默认值对象一样，必须在绑定之后才能起作用。绑定之后的数据库对象，就如同定义了 CHECK 约束一样，在插入或修改数据时检验新数据。

规则的绑定需要使用系统存储过程 sp_bindrule，具体语法如下所示。

```
USE 数据库名
GO
sp_bindrule 规则名 表名.列名
[,@futureonly=< futureonly_flag >]
```

在上述语法中，“[,@futureonly=< futureonly_flag >]”参数将规则绑定到用户自定义数据类型

时使用。如果 futureonly_flag 为空，则该数据类型已有的数据将不受限制。如果不指定 futureonly，则该规则将绑定到所有使用该数据类型的列上并对已有的数据进行验证。

【练习 30】

将 medicine 数据库中的 CheckTime 规则绑定到 EmployeerInfo 表的 EmployeerWorkday 字段，实现语句如下。

```
USE medicine
GO
sp_bindrule CheckTime,'EmployeerInfo.EmployeerWorkday'
```

执行上述语句后会在【消息】区域中提示“已将规则绑定到表的列”，如图 5-13 所示。绑定完成后，在列属性中也可以看到。

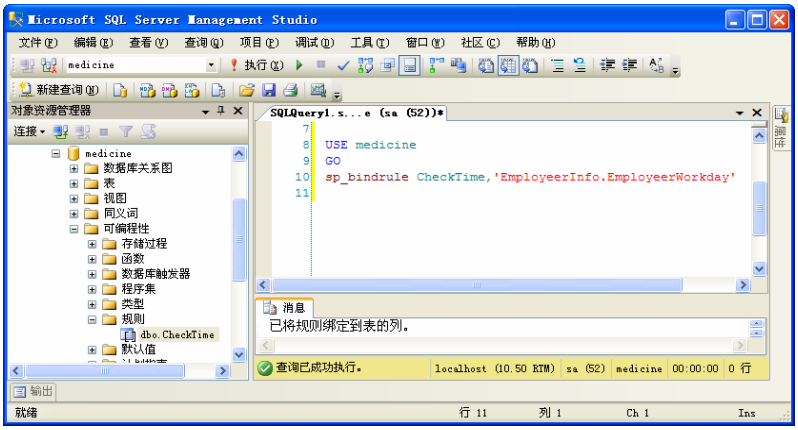

图 5-13　规则绑定

因为规则不是针对某一列或某个用户自定义数据类型，所以在该数据库对象不再需要使用规则的时候，可以取消对规则的绑定而不需要直接删除规则。取消对规则的绑定需要使用系统存储过程 sp_unbindrule，语法如下所示。

```
sp_unbindrule 表名.字段名
[,@futureonly=< futureonly_flag >]
```

【练习 31】

将 medicine 数据库中 EmployeerInfo 表的 EmployeerWorkday 字段解除规则绑定，实现语句如下。

```
USE medicine
GO
sp_unbindrule 'EmployeerInfo.EmployeerWorkday'
```

5.4.3　查看规则

使用存储过程 sp_help 查看规则，包括规则名称、所有者和创建时间等，具体语法如下。

```
sp_help [规则名]
```

在不写规则名的情况下，系统将会指定数据库中所有规则、索引、约束等查询，这个结果里面没有创建时间。

【练习 32】

查询 medicine 数据库中 CheckTime 规则的信息，实现语句如下。

```
USE medicine
GO
sp_help CheckTime
```

执行结果如图 5-14 所示。

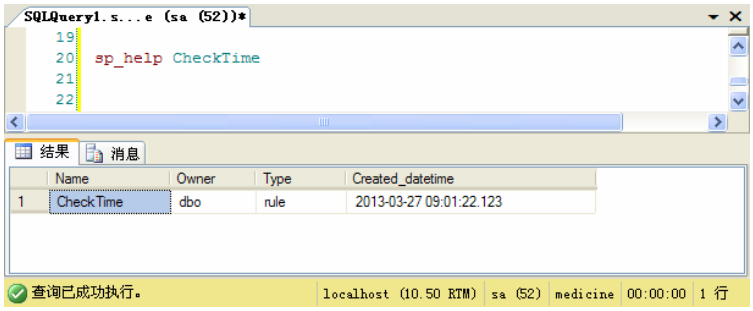

图 5-14　查看规则

【练习 33】

查询 medicine 数据库中 CheckTime 规则的定义，使用存储过程 sp_helptext 实现语句如下。

```
USE medicine
GO
sp_helptext CheckTime
```

执行结果如图 5-15 所示。

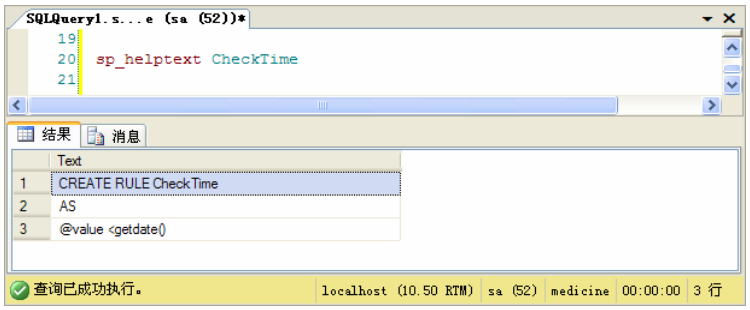

图 5-15　查询规则的定义

5.4.4　删除规则

不使用的规则可以使用 DROP RULE 语句删除，具体语法如下。

```
DROP RULE 规则名
```

【练习 34】

删除 medicine 数据库中的 CheckTime 规则，实现语句如下。

```
USE medicine
GO
DROP RULE CheckTime
```

5.5 实例应用：维护订单数据完整性

5.5.1 实例目标

网购已经被越来越多的用户接受和喜爱，可以选择的网购网站也越来越多。在网购网站中订单是最重要的核心功能模块，它记录了用户的购买信息以及购买的商品信息，因此订单数据的完整性和有效性是每个网购网站必须要解决的。

学习本课的内容之后，读者可以从数据库方面制订维护订单数据完整性的计划。这个计划主要体现在三个方面，即在数据库中保存哪些订单信息，这些信息划分为哪些数据表，每个数据表的数据如何约束。

5.5.2 技术分析

以一个简单的网购网站为例，将订单划分为两个表进行存储。第一个表用于存储订单的基本信息，详细描述如表 5-1 所示。

表 5-1 订单表

列 名	数 据 类 型	是否允许为空	备 注
流水号	int	否	自动编号
订单号	varchar(12)	否	主键
客户名称	varchar(50)	否	
联系电话	varchar(11)	是	必须为数字
收货地址	varchar(100)	否	
下单日期	datetime	否	默认为当前日期
物流名称	varchar(50)	是	
物流费用	int	是	默认为 10
物流编号	varchar(8)	否	
付款方式	varchar(50)	是	现金|支票|货到付款

第二个表用于存储订单对应的商品明细，详细描述如表 5-2 所示。

表 5-2 订单明细表

列 名	数 据 类 型	是否允许为空	备 注
流水号	int	否	主键、自动编号
订单号	varchar(12)	否	外键
数量	int	否	大于 0
价格	float	否	大于 0

5.5.3 实现步骤

（1）根据表 5-1 和表 5-2 对列的分析，下面开始创建表，并同时对列的约束进行设置。如下所示为订单表的创建语句。

```
CREATE TABLE 订单表
(
```

```
id int IDENTITY(1,1),
订单号 varchar(12) NOT NULL PRIMARY KEY,
客户名称 varchar(50) NOT NULL,
联系电话 varchar(11),
下单日期 datetime NOT NULL DEFAULT getdate(),
收货地址 varchar(100) NOT NULL,
物流名称 varchar(50),
物流费用 int DEFAULT 10,
物流编号 varchar(8) NOT NULL,
付款方式 varchar(50)
CONSTRAINT CheckPayment
CHECK (付款方式 in('现金','支票','货到付款'))
)
```

（2）联系电话必须为 11 位的数字，且第 1 位数字不为 0。为了实现这个限制以下创建了一个名为 phoneNum 的规则进行验证。

```
CREATE RULE phoneNum
AS
@value like'[1-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
```

（3）将 phoneNum 规则绑定到订单表的联系电话列。

```
sp_bindrule phoneNum,'订单表.联系电话'
```

（4）创建订单明细表，实现语句如下。

```
CREATE TABLE 订单明细表
(
id int PRIMARY KEY IDENTITY(1,1),
订单号 varchar(12) NOT NULL FOREIGN KEY REFERENCES 订单表(订单号),
数量 int NOT NULL,
价格 float NOT NULL
)
```

（5）由于需要多次用于验证是否大于 0，所以创建了一个规则来实现。

```
CREATE RULE validNumber
AS
@Number>0
```

（6）将上面创建的规则依次绑定到数量列和价格列。

```
sp_bindrule validNumber,'订单明细表.数量'
GO
sp_bindrule validNumber,'订单明细表.价格'
```

（7）上面语句的执行完成后，整个实例就会完成。接下来可以向表中添加数据以验证各个约束的有效性。

5.6 拓展训练

1．为学生信息表设计约束

创建一个“学生信息”表，该表包括列有“编号”、“学生编号”、“学生姓名”、“性别”、“政治面貌”和“家庭住址”，然后对该表应用如下约束。

- 对“编号”列使用自动编号。
- 将“学生编号”列设置为主键。
- 为“学生姓名”列和“家庭住址”列使用惟一性约束。
- 将“党员”作为“政治面貌”列的默认值。
- 检查“性别”列的有效性

2．使用规则约束完整性

创建一个“客户信息”表，该表包括列有“客户编号”、“客户姓名”、“联系电话”和“所在城市”，然后使用规则完成如下约束：

（1）创建一个名为“所在城市_rule”的规则，限定输入的值必须是“北京市”、“广州市”、“南京市”、“上海市”、“深圳市”、“天津市”、“西安市”、“郑州市”之一。

（2）使用 sp_bindrule 语句将规则绑定到“所在城市”列。

（3）添加数据测试规则的有效性。

（4）使用 sp_unbindrule 语句解除规则的绑定。

（5）删除“所在城市_rule”规则。

5.7 课后练习

一、填空题

1. 数据完整性分为四类，分别是实体完整性、域完整性、_____和用户定义的完整性。

2. _____是关系数据库中增强表之间参照完整性的主要机制。

3. _____的使用方法与规则非常相似，也可以通过和表列或者用户自定义数据类型进行绑定，对新增加的数据进行约束。

4. 创建默认值所使用的命令是_____。

5. _____是数据库对象之一，它的作用与 CHECK 约束的部分功能相同，在向表的某列插入或更新数据时，用它来限制输入新值的取值范围。

6. 创建规则的命令是_____。

二、选择题

1. 下列约束不属于域完整的是_____。

A. 验证约束

B. 外键约束

C. 自动编号

D. 默认值

2. 关于约束，下列哪种说法是正确的？_____

A. 表数据的完整性用表约束就足够了

B. 自动编号的列数据都是有固定差值的

C. 一个列只能有一个 CHECK 约束

D. UNIQUE 约束列可以为 NULL

3. 下列关于规则说法错误的是_____。

A. CHECK 约束是用 DREATE TABLE 语句在建表时指定的，而规则需要作为单独的数据库对象来实现

B. 在一列上只能使用一个规则，但可以使用多个 CHECK 约束

C. 规则可以应用于多个示例，还可以应用于用户自定义的数据类型，而 CHECK 约束只能应用于它定义的行

D. 规则是实现域完整性的方法之一，它用来验证一个数据库中的数据是否处于一个指定的值域范围内

4. 下列说法正确的是_____。

A. 规则的修改需要先删除，再重新创建

B. 新建的列默认为 NOT NULL

C. CHECK 约束修改需要先删除，再重建

D. 默认值可以是任意有返回值的函数

三、简答题

1. 简述在列中使用空和非空的意义。

2. 简述 PRIMARY KEY 约束所受到的限制。

3. 简述创建 FOREIGN KEY 约束时应遵循的基本原则。

4. 简述规则与 CHECK 约束有哪些不同?

5. 简述默认值约束与默认值对象有何区别。