项目8 实现数据完整性

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训

【能力目标】

- 能描述数据完整性的含义及分类
- 学会使用CHECK约束、RULE规则、DEFAULT默认值约束、DEFAULT默认值对象保证列数据完整性（即域完整性）
- 学会使用索引、PRIMARY KEY主键约束、UNIQUE唯一约束或IDENTITY属性来保证行数据完整性（即实体完整性）
- 学会使用从表的FOREIGN KEY外键约束与主表的定义主键PRIMARY KEY或唯一键UNIQUE约束（不允许为空）实现主表与从表之间的参照完整性

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训

【项目描述】

为XS数据库创建CHECK约束、RULE规则、DEFAULT默认值约束、DEFAULT默认值对象、索引、PRIMARY KEY主键约束、UNIQUE唯一约束、FOREIGN KEY外键约束实现数据完整性保护。

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训

【项目分析】

项目4在学生数据库XS中建立了数据表，在向表中输入数据时，由于种种原因，有时会输入无效或错误的信息。比如：对不同的学生输入了相同的学号；性别字段的值输入了非法数据；相同的数据行被多次输入；学生成绩表中出现了学生档案表中不存在的学号等。之所以会出现这些错误信息，是因为没有实现数据完整性设计。为避免此类情况发生，本项目主要介绍如何通过实施数据完整性来解决上述问题。以此保证数据输入的正确性、一致性和可靠性。

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训

【任务设置】

任务1……实现域完整性

任务2……实现实体完整性

任务3……实现参照完整性

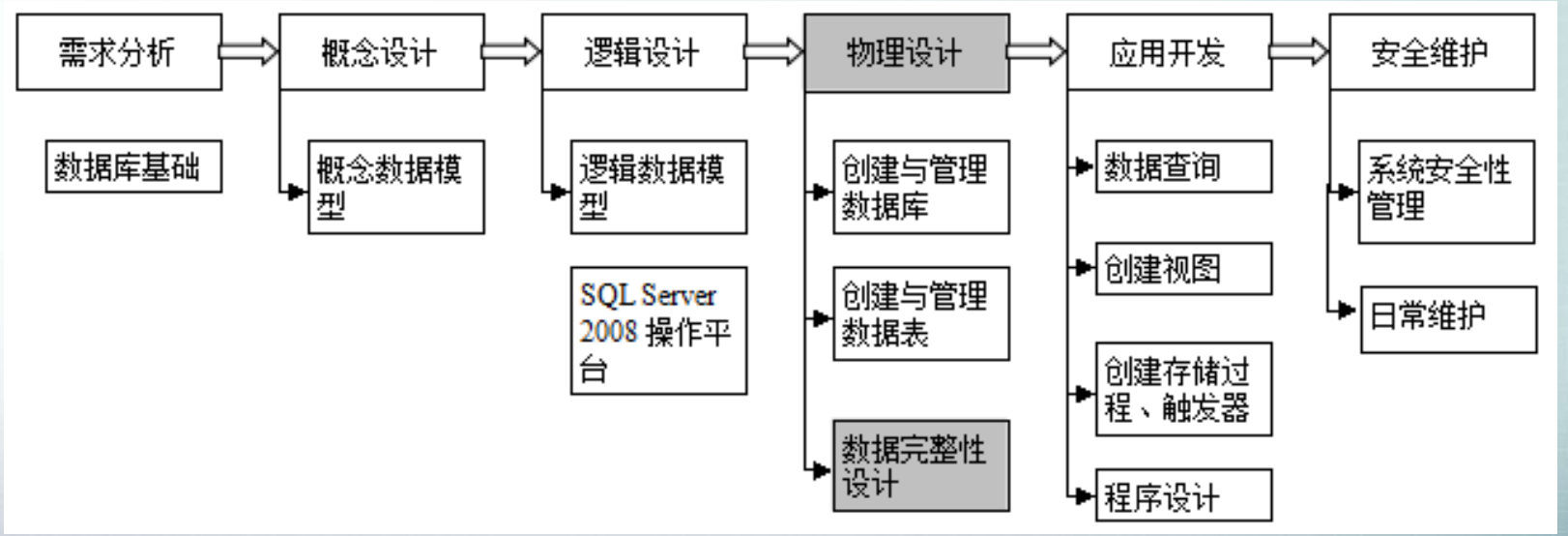
实训八……实现sale数据库完整性

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训

【项目定位】

数据库系统开发



任务1 实现域完整性

【任务目标】

- 能阐述数据完整性的概念
- 能阐述数据完整性的准确分类
- 学会使用CHECK约束、RULE规则、DEFAULT默认值约束、DEFAULT默认值对象实现域完整性

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训

【任务描述】

按需求在XS数据库完成以下域完整性相关的操作。

1. 使用CREATE语句创建表book（书号char(6)、书名char(20)、类型char(20)、价格int）字段。给“价格”字段定义一个名为max_price的check约束 ，使得价格不超过200。
2. 定义一个名为type_rule的规则，并绑定到book表的“类型”字段，使得“类型”字段只接受该规则中列出的值（计算机类、科普类、文学类、自然科学）。
3. 解除第2题中的规则绑定关系。
4. 删除第2题中的规则。
5. 修改book表,为“类型”字段设置一个默认值约束，约束名为BookType，默认值为“NEW BOOK”。
6. 从book表中删除默认值约束BookType。

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训

7. 定义默认值对象Day，取值为getdate()。
8. 修改XSDA表，向学生表中添加字段“入学时间”。
9. 将默认值对象Day绑定到XSDA表的“入学时间”字段。
10. 定义一个用户定义数据类型：Today。
11. 将默认值对象Day绑定到用户自定义数据类型ToDay。
12. 修改book表，向book表中添加字段“购买日期”，定义类型为ToDay。
13. 解除默认值对象Day与XSDA表“入学时间”、自定义数据类型Today的绑定关系。
14. 删除默认值对象Day。
15. 阐述数据完整性的分类，以及每类完整性都由哪些约束实现。

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训

【任务分析】

练习域完整性的实现，包括：CHECK约束及默认值约束的创建，规则及默认值对象的定义、绑定、解除绑定、删除。特别要注意：规则和默认值对象，使用它们要首先定义，然后绑定到列或用户定义数据类型；不需要时要先解除绑定，然后删除规则。

任务1-1 认知数据完整性概念及分类

数据完整性就是用于保证数据库中的数据在逻辑上的一致性、正确性和可靠性。强制数据完整性可确保数据库中的数据质量。数据完整性一般包括3种类型：域完整性、实体完整性、参照完整性。

1．域完整性

域完整性又称为列完整性，指给定列输入的有效性，即保证指定列的数据具有正确的数据类型、格式和有效的数据范围。实现域完整性可通过定义相应的CHECK约束、默认值约束、默认值对象、规则对象等方法来实现，另外，通过为表的列定义数据类型和NOT NULL也可以实现域完整性。

例如，KCXX表中每门课程的学分应在0~10之间，为了对学分这一数据项输入的数据范围进行限制，可以在定义KCXX表结构的同时通过定义学分的CHECK约束来实现。

2. 实体完整性

实体完整性又称为行的完整性，是用于保证数据表中每一个特定实体的记录都是唯一的。通过索引、UNIQUE约束、PRIMARY KEY约束或IDENTITY属性可以实现数据的实体完整性。

例如，对于XSDA表，学号作为主键，每一个学生的学号都能唯一地标识该学生对应的行记录信息，那么在输入数据时，就不能有相同学号的行记录，通过对学号字段建立PRIMARY KEY约束可以实现XSDA表的实体完整性。

3. 参照完整性

当增加、修改或删除数据表中的记录时，可以借助参照完整性来保证相关联表之间数据的一致性。参照完整性可以保证主表中的数据与从表中数据的一致性。在SQL SERVER 2008中，参照完整性是通过定义外键与主键之间或外键与唯一键之间的对应关系来实现的。参照完整性确保同一键值在所有表中一致。

例如，对于XS数据库中的XSDA表中的每个学生的学号，在XSCJ表中都有相关的课程成绩记录，将XSDA表作为主表，学号字段定义为主键，XSCJ表作为从表，表中的学号字段定义为外键，从而建立主表和从表之间的联系实现参照完整性。

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训

XSDA表和XSCJ表的对应关系如表8-1、表8-2所示。

表 8-1 XSDA 表

学号(主键)	姓 名	性 别	系 名	总学分
200801	王红	女	信息	60
200802	刘林	男	信息	54

表 8-2 XSCJ 表

学号(外键)	课程编号	成 绩
200801	104	81
200801	108	77
200801	202	89
200801	207	90
200802	104	92
200802	108	95
200802	202	93
200802	207	90

如果定义了两个表之间的参照完整性，则要求：

（1）从表不能引用不存在的键值。例如：对于XSCJ表的行记录中出现的学号必须是XSDA表中已经存在的学号。

（2）如果主表中的键值更改了，那么在整个数据库中，对从表中该键值的所有引用要进行一致的更改。例如：如果XSDA表中的某一学号修改了，XSCJ表中所有对应学号也要进行相应的修改。

（3）如果主表中没有关联的记录，则不能将记录添加到从表中。

（4）如果要删除主表中的某一记录，应先删除从表中与该记录匹配的相关记录。

任务1-2 CHECK约束

CHECK约束实际上是字段输入内容的验证规则，表示一个字段的输入内容必须满足CHECK约束的条件，若不满足，则数据无法正常输入。

CHECK约束可以作为表定义的一部分在创建表时创建，也可以添加到现有表中。表和列可以包含多个CHECK约束。允许修改或删除现有的CHECK约束。

1. 使用SQL Server Management Studio定义CHECK约束

在XS数据库的XSCJ表中，学生每门课程的成绩一般在0~100的范围内，如果对用户的输入数据要施加这一限制，可以按照如下步骤进行操作。

（1）启动SQL Server Management Studio，打开XSCJ表的表设计器窗口，单击鼠标右键，出现如图8-1所示的快捷菜单。

（2）选择【CHECK 约束】选项，进入如图8-2所示的“CHECK约束”属性窗口。

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训

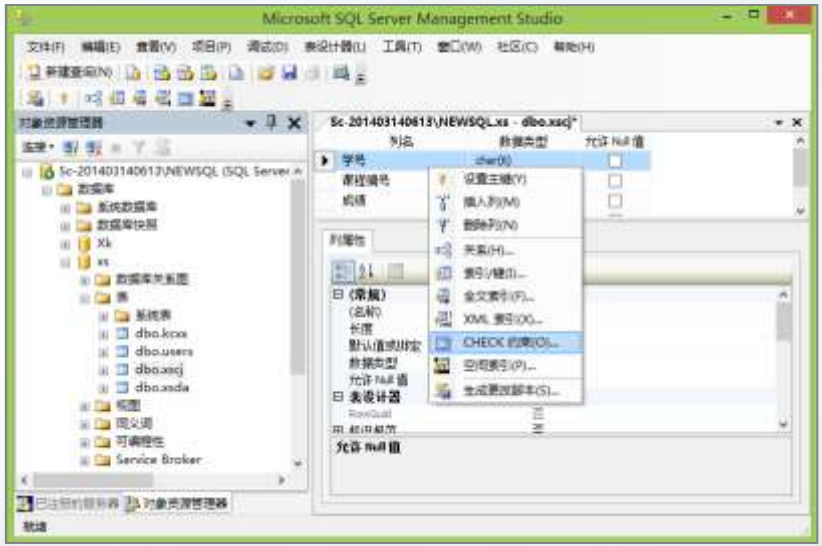


图8-1 表设计器快捷菜单

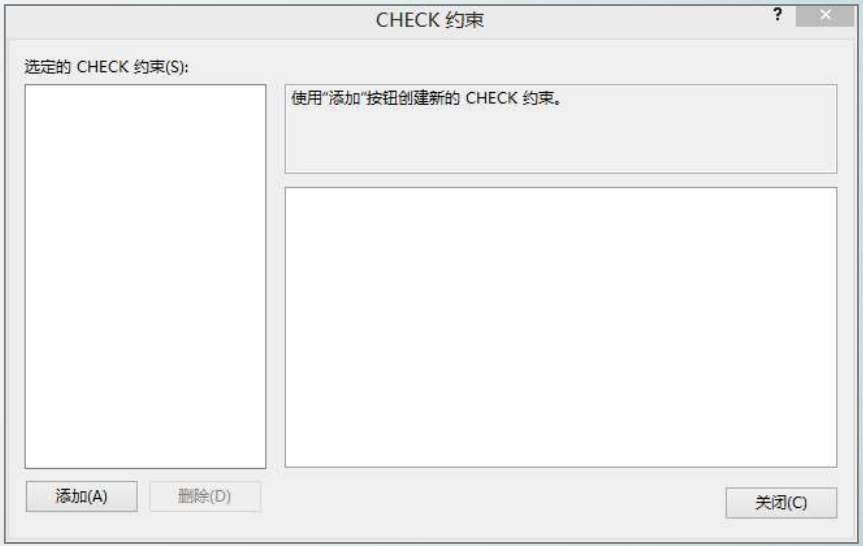


图8-2 “CHECK约束”属性窗口

（3）单击【添加】按钮，进入如图8-3 所示的的“CHECK约束”设置窗口，在【选定的CHECK约束】框中显示由系统分配的新约束名。名称以“CK_”开始，后跟表名，可以修改此约束名。在【表达式】框中，可以直接输入约束表达式“成绩>=0 and 成绩<=100”，如图8-3所示。也可单击【表达式】框右侧的“浏览”按钮，弹出如图8-4所示的【CHECK约束表达式】对话框，可以在其中编辑表达式。

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训

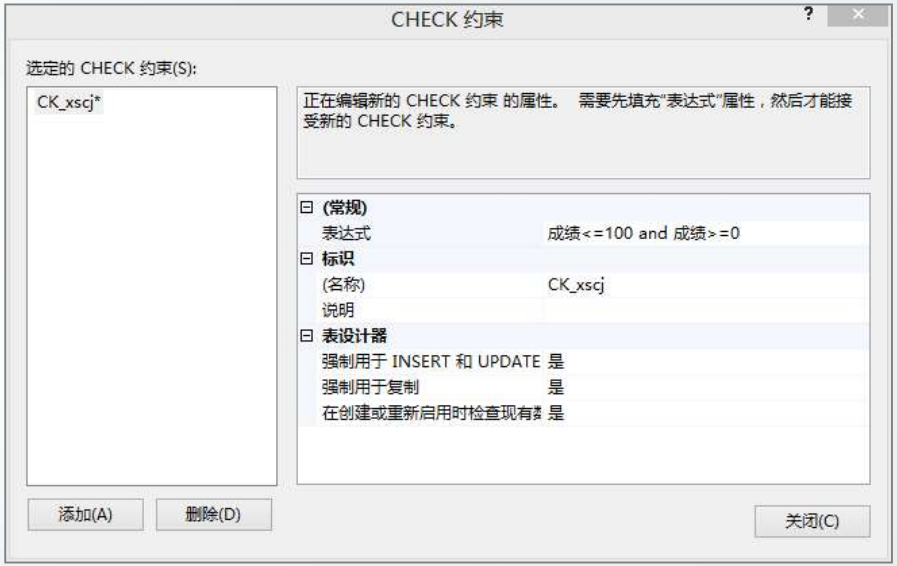


图8-3 “CHECK约束”设置窗口

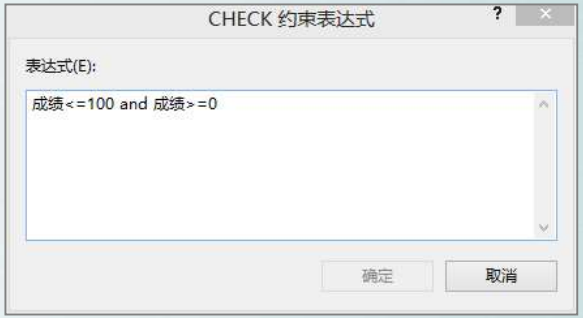


图8-4 “CHECK约束表达式编辑”窗口

（4）如果允许创建的约束用于强制INSERT和UPDATE以及检查原有数据等情况，可以在图8-3所示窗口右下方的相应选项中选择“是”。最后，单击【关闭】按钮，则完成了CHECK约束的设置。（如果检查时，数据冲突，无法成功保存设置）

按上述步骤创建约束后，输入数据时如果成绩不在0~100的范围内，系统将报告错误。

如果要删除上述约束，只需进入如图8-3所示的CHECK约束设置窗口，在【选定的CHECK约束】框中选择要删除的约束，然后单击【删除】按钮即可。

注意：对于TimeStamp 和Identity两种类型的字段不能定义CHECK约束。

2. 使用T-SQL语句在创建表时定义CHECK约束

使用SQL语句创建表结构时，可以定义CHECK约束。

语法格式：

```
CREATE TABLE table_name                                    /*指定表名
(column_name  datatype  NOT NULL | NULL                    /*定义列名、数据类型、是否空值
[[CONSTRAINT check_name] CHECK (logical_expression)][,…n])  /*定义CHECK约束
```

说明：关键字CHECK表示定义CHECK约束，其后的logical_expression是逻辑表达式，称为CHECK约束表达式。

【例8-1】在XS数据库中创建学生信息表XSXX并定义CHECK约束。

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训

```
USE XS
CREATE TABLE XSXX
(
学号 char(6),
姓名 char(8),
性别 char(2) CHECK (性别 IN ('男','女')),
入学日期 datetime
)
GO
```

这两种语法的区别？

```
USE XS
CREATE TABLE XSXX
(
学号 char(6),
姓名 char(8),
性别 char(2) constraint ck_sex  CHECK (性别
IN ('男','女')),
入学日期 datetime
)
GO
```

3. 使用T-SQL语句在修改表时定义CHECK约束

对于已经存在的表，也可以定义CHECK约束。在现有表中添加CHECK约束时，该约束可以仅作用于新数据，也可以同时作用于已有的数据。

语法格式：

```
ALTER TABLE table_name  [WITH CHECK | WITH NOCHECK]
ADD CONSTRAINT check_name CHECK (logical_expression)
```

说明：关键字ADD CONSTRAINT表示在已经定义的table_name表中增加一个约束定义，约束名由check_name指定，约束表达式为logical_expression。WITH CHECK选项表示CHECK约束同时作用于已有数据和新数据；当省略该选项，取默认设置时，也表示CHECK约束同时作用于已有数据和新数据；WITH NOCHECK选项表示CHECK约束仅作用于新数据，对已有数据不强制约束检查。

【例8-2】通过修改XS数据库中的XSCJ表，增加成绩字段的CHECK约束。

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训

```
USE XS
GO
ALTER TABLE xscj
ADD CONSTRAINT ck_cj check(成绩>=0 and 成绩<=100)
GO
```

4. 使用T-SQL语句删除CHECK约束

语法格式：

```
ALTER TABLE table_name
DROP CONSTRAINT check_name
```

说明：在table_name指定的表中，删除名为check_name的约束。

【例8-3】删除XSCJ表中的成绩字段的CHECK约束。

```
alter table xscj
drop constraint ck_cj
```

任务1-3 规则

规则是保证域完整性的主要手段，它类似于CHECK约束。与CHECK约束相比，其执行功能相同。CHECK约束是使用ALTER TABLE 或CREATE TABLE的CHECK关键字创建的，是对列中的值进行限制的首选标准方法（可以对一列或多列定义多个约束）。

规则是一种数据库对象，可以绑定到一列或多个列上，还可以绑定到用户定义数据类型上，规则定义之后可以反复使用。

列或用户定义数据类型只能有一个绑定的规则。（即使有多个绑定，以最后绑定为准）但是，列可以同时具有规则和多个CHECK约束。

规则作为独立的数据库对象，使用时要首先定义，规则定义后只有将其绑定到列或用户定义数据类型后才能生效；不需要时可以先解除绑定，然后删除规则。

1. 定义规则

规则只能利用T-SQL语句定义。

语法格式：

```
CREATE RULE rule AS condition_expression
```

说明：参数rule是定义的新规则名，规则名必须符合标识符命名规则；参数condition_expression是规则的条件表达式，该表达式可为WHERE子句中任何有效的表达式，但规则表达式中不能包含列或其他数据库对象，可以包含不引用数据库对象的内置函数，在condition_expression条件表达式中包含一个局部变量，使用UPDATE或INSERT语句修改或插入值时，该表达式用于对规则关联的列值进行约束。定义规则时，一般使用局部变量表示UPDATE或INSERT语句输入的值。

另外有如下几点需要说明：

定义的规则对先前已存在于数据库中的数据无效。（即不会检查之前存在的数据）

在单个批处理中，CREATE RULE语句不能与其他T-SQL语句组合使用。

规则表达式的类型必须与列的数据类型兼容，不能将规则绑定到text，image，timestamp列。要用单引号将字符和日期常量括起来，在十六进制常量前加0x。

对于用户定义数据类型，当在该类型的数据列中插入值，或更新该类型的数据列时，绑定到该类型的规则才会激活。规则不检验变量，所以在向用户定义数据类型的变量赋值时，不能与列绑定的规则冲突。

如果列同时有默认值和规则与之关联，则默认值必须满足规则的定义，与规则冲突的默认值不能插入列。

2. 绑定规则

规则定义之后，需要将其绑定到列或用户定义数据类型后才能起作用。当向绑定了规则的列或使用绑定规则的用户定义数据类型的所有列插入或更新数据时，新的数据必须符合规则。绑定规则一般是通过T-SQL语句实现，在SQL Server Management Studio下规则只能绑定到用户定义数据类型，方法类似于默认值的绑定，在此不再赘述。

语法格式：

```
sp_bindrule [@rulename=]' rule' ,
[@objname=]' object_name'
[,[@futureonly=]' futureonly_flag' ]
```

说明：参数rule为CREATE RULE语句创建的规则名，要用单引号括起来；参数object_name是绑定到规则的列或用户定义数据类型，如果object_name采用“表名.字段名”的格式，则认为绑定到表的列，否则绑定到用户定义数据类型；参数futureonly_flag仅当将规则绑定到用户定义数据类型时才使用。

futureonly_flag的默认值为 NULL，如果futureonly_flag为NULL，那么新规则将绑定到用户定义数据类型的每一列，条件是此数据类型当前无规则或者使用用户定义数据类型的现有规则；如果futureonly_flag设置为’futureonly’，那么这个规则只可用于用户定义数据类型的新列，现有列将不继承新规则。

【例8-4】定义一个规则，用以限制数据输入模式。比如：对于KCXX表的课程编号列，如果规定课程编号的第1位代表开课学期(输入时只能选1~6)，后2位代表课程序号(输入时只能选0~9)。按此要求定义一个规则，并绑定到KCXX表的课程编号列，用于限制课程编号的输入。

```
create rule kcbh_rule as @range like '[1-6][0-9][0-9]'
go
exec sp_bindrule 'kcbh_rule','kcxx.课程编号'

INSERT kcxx
values (‘7604’,‘计算机文化基础’,‘1’,‘60’,‘3’)  ‘用于验证
```

【例8-5】定义一个规则，用以限制数据输入的范围。比如：对于XSCJ表的成绩列，规定成绩只能输入0~100之间的数。按此要求定义一个规则，并绑定到XSCJ表的成绩列，用于限制成绩的输入。

```
create rule cj_rule as @score>=0 and @score <=100
go
exec sp_bindrule 'cj_rule','xscj.成绩'
```

【例8-6】定义一个规则，用以限制输入到该规则所绑定的列中的值只能是该规则中列出的值。比如：对于XSDA表的系名列，规定系名只能输入“信息”、“管理”、“机械”、“电气”中的值。按此要求定义一个规则，并绑定到XSDA表的系名列，用于限制系名的输入。

```
create rule xm_rule as @name in('信息','管理','机械','电气')
go
exec sp_bindrule 'xm_rule','xsda.系名'
```

【例8-7】如下程序定义一个用户定义数据类型，然后将前面定义的规则“kcbh_rule”绑定到用户定义数据类型course_num上，最后创建表KCXX1，其课程编号的数据类型为course_num。

```
exec sp_addtype 'course_num','char(3)','not null'
exec sp_bindrule 'kcbh_rule','course_num'
go
create table kcxx1
(课程编号 course_num,
 课程名称 char(16) not null,
 学分 tinyint)
```

3. 解除绑定

如果要删除规则，首先应先解除规则与列或用户定义数据类型之间的绑定关系，然后才能删除规则。

（1）使用SQL Server Management Studio解除绑定

使用SQL Server Management Studio解除绑定的方法，类似于绑定操作。只适用于用户定义数据类型，方法如下。

① 打开SQL Server Management Studio，展开“XS数据库”，展开【可编程性】|【类型】|【用户定义数据类型】。

② 选中要解除绑定的类型，单击右键选择【属性】，进入“用户定义数据类型”界面，删除规则，单击【确定】按钮完成。

（2）使用T-SQL语句解除绑定

解除绑定主要使用T-SQL语句实现。

语法格式：

sp_unbindrule [@objname=]’ object_name’

[,[@futureonly=]’ futureonly_flag’]

说明：参数object_name用于指定解除规则绑定的列或者用户定义数据类型名。如果object_name是“表名.字段名”格式，则object_name为表中的列，否则object_name为用户定义数据类型。为用户定义数据类型解除规则绑定时，所有属于该数据类型并具有相同规则的列也同时解除规则绑定。对属于该数据类型的列，如果其规则直接绑定到列上，则该列不受影响。参数futureonly_flag仅用于解除用户定义数据类型规则的绑定，当参数futureonly_flag取值为futureonly时，规则仍然对现有的属于该数据类型的列有效。

4. 删除规则

在解除列或用户定义数据类型与规则之间的绑定关系后，就可以删除规则了。

语法格式：

```
DROP RULE {rule}[,…n]
```

说明：参数rule指定删除的规则名，可以包含规则所有者名；参数n表示可以指定多个规则同时删除。

【例8-8】解除课程编号列与规则kcbh_rule之间的绑定关系。

【例8-9】解除用户定义数据类型course_num与规则kcbh_rule之间的绑定关系，并删除规则kcbh_rule。

执行DROP RULE命令时，如果规则没有绑定到列或用户定义数据类型，将显示命令成功完成信息；如果有绑定，则显示无法删除信息。解决方法是先解除绑定，再删除。可以通过对象资源管理器找到规则名，单击右键选择“查看依赖关系”，即可看到有哪些列或用户定义数据类型绑定了规则。

任务1-4 默认值约束及默认值对象

对于某些字段，可以为其定义默认值，以方便用户的使用。一个字段的默认值的建立可通过如下两种方式实现：

（1）在创建表或修改表时，定义默认值约束。

（2）先定义默认值对象，然后将该对象绑定到表的相应字段或用户定义数据类型上。

1．默认值约束的定义及删除

默认值约束是在用户未提供某些列的数据时，数据库系统为用户提供的默认值，从而简化应用程序代码和提高系统性能。

表的每一列都可包含一个默认值定义。可以修改或删除现有的默认值定义。默认值必须与默认定义适用的列的数据类型相一致，每一列只能定义一个默认值。

（1）定义默认值约束

在创建表或修改表时，可以定义一个字段的默认值约束。默认值约束的定义可以通过SQL Server Management Studio实现（前面的章节已经介绍过，这里不再重述），也可以通过T-SQL语句来实现，下面介绍如何使用T-SQL语句定义一个字段的默认值约束。

①在创建表时定义默认值约束

语法格式：

```
CREATE TABLE table_name                          /*指定表名
(column_name datatype NOT NULL | NULL      /*定义列名、数据类型、是否空值
[CONSTRAINT default_name][DEFAULT constraint_expression] [,…n])
 /*定义默认值约束
```

说明：table_name为创建的表名，column_name为列名，datatype为对应列的数据类型；DEFAULT关键字表示其后的constraint_expression表达式为默认值约束表达式，此表达式只能是常量、系统函数或NULL。对于timestamp或带IDENTITY属性的字段不能定义默认值约束。参数n表示可定义多个数据字段。

【例8-10】在创建表时定义一个字段的默认值约束。

②在修改表时定义默认值约束

在修改表时定义默认值约束有两种情况：一是对表中已存在的列添加默认值约束；二是对表增加新列时，定义新列的默认值约束。

语法格式1：

```
ALTER TABLE table_name                                    /*指定表名
ADD [CONSTRAINT default_name][DEFAULT constraint_expression] FOR
column_name [,…n]                                         /*对已存在的列添加默认值约束
```

说明：此格式是对表中已存在的列添加默认值约束。

语法格式2：

```
ALTER TABLE table_name                          /*指定表名
ADD column_name datatype NOT NULL | NULL        /*为增加的新列定义列名、数据类型、是否空值
[CONSTRAINT default_name][DEFAULT constraint_expression] [,…n] /*定义默认值约束
```

说明：此格式是对表增加新列时定义默认值约束。

【例8-11】在修改表时为XSDA表中已存在的列“民族”定义默认值约束“汉”。

```
alter table xsda
add default '汉' for 民族
```

【例8-12】在修改表时通过增加一个新字段，定义默认值约束。

```
alter table xsda1
add 政治面貌 char(4) not null constraint zzmmdflt default '团员
```

（2）删除默认值约束

默认值约束可以在SQL Server Management Studio中删除。（该方法需要在表设计器中删除）如果已知一个默认值约束的约束名，也可以通过T-SQL语句进行删除，其使用方法如下例所示。

【例8-13】删除上例定义的默认值约束。

```
ALTER TABLE XSDA1
  DROP CONSTRAINT zzmmdflt
```

2. 默认值对象的定义、使用及删除

默认值对象是一种数据库对象，可以被绑定到一个或多个列上，还可以绑定到用户定义数据类型上。当某个默认值对象定义后，可以反复使用。当向表中插入数据时，如果绑定有默认值对象的列或者用户定义数据类型没有明确提供值，那么就将以默认值指定的数据插入。定义的默认值对象必须与所绑定列的数据类型一致。

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训

默认值对象的执行与前面所讲的默认值约束功能相同，默认值约束的定义和表存储在一起，当删除表时，将自动删除默认值约束。默认值约束是限制列数据的首选标准方法。然而，当在多个列中，特别是不同表中的列中多次使用默认值时，适合采用默认值对象技术。要使用默认值对象，首先要定义默认值对象，然后将其绑定到指定列或用户定义数据类型上。当取消默认值时，要先解除绑定，再删除默认值。

（1）定义默认值对象

默认值对象的定义只能通过T-SQL语句实现。（同rule）

语法格式：

```
CREATE DEFAULT default
AS constaint_expression
```

说明：CREATE DEFAULT关键字表示创建一个名为default的默认值对象，默认值对象名必须符合标识符规则，可以包含默认值对象所有者名。约束表达式constaint_expression只能是常量表达式（不能包含字段名或其他数据库对象的名称），可以含有常量、内置函数，字符和日期常量用单引号括起来；货币、整数和浮点常量不需要使用引号。十六进制数据必须以0x开头，货币数据必须以美元符号（$）开头。默认值对象必须与列数据类型兼容。

【例8-14】定义默认值对象，设置其默认值为“汉”。

```
create default mz_default as '汉'
```

（2）绑定默认值对象

默认值对象定义之后，需要将其绑定到列或用户定义数据类型上才能生效。

①使用SQL Server Management Studio绑定默认值对象

将【例8-14】定义的默认值对象“mz_default”，绑定到XSDA表的民族字段上，方法如下。（参考P218）

②使用T-SQL语句绑定默认值对象

创建默认值对象后，要使其起作用，应使用sp_bindefault存储过程将其绑定到列或用户定义数据类型上。

语法格式：

```
sp_bindefault[@defname=]’ default’ ,
[@objname=]’ object_name’
[,[@futureonly=]’ futureonly flag’ ]
```

说明：参数default指定由CREATE DEFAULT语句创建的默认值对象名，要用单引号括起来；参数object_name指定准备绑定默认值对象的表的列名（格式为：表名.字段名）或用户定义的数据类型名，object_name要用单引号括起来。不能将默认值对象绑定到timestamp数据类型的列、带IDENTITY属性的列或者已经有DEFAULT约束的列；参数futureonly_flag仅在将默认值对象绑定到用户定义数据类型时才使用，默认值为 NULL。当futureonly_flag的值为futureonly时，表示在此之前，该数据类型关联的列不继承该默认值对象的值；如果 futureonly_flag为 NULL，那么新默认值将绑定到用户定义数据类型的任一列，条件是此数据类型当前无默认值或者使用用户定义数据类型的现有默认值。

【例8-15】对于XSDA表中的民族字段通过定义默认值对象，实现默认值设置为“汉”。

```
use xs
 go
 create default mz_default as '汉'
 go
 use xs
 exec sp_bindefault 'mz_default','xsda.民族'
 go
```

【例8-16】在XS数据库中创建名为rxdate的默认值对象（取值为当前系统日期），然后将其绑定到XSXX表（例8-1中创建）的入学日期列。

```
use xs
go
create default rxdate as getdate()
go
use xs
exec sp_bindefault 'rxdate','xsxx.入学日期'
go
```

【例8-17】在XS数据库中定义一个名为sex的用户定义数据类型，然后定义默认值对象xb_default，并将其绑定到用户定义数据类型sex中。

```
use xs
 go
 exec sp_addtype sex,'char(2)',null
 go
 create default xb_default as '男'
 go
 use xs
 go
 exec sp_bindefault 'xb_default','sex'
 go
```

（3）解除绑定

如果要删除一个默认值对象，首先应解除默认值对象与用户定义数据类型及表字段的绑定关系，然后才能删除该默认值对象。

绑定关系的解除可以在SQL Server Management Studio中实现，也可以通过SQL语句实现。

①使用SQL Server Management Studio解除绑定关系

在SQL Server Management Studio中解除绑定的方法，类似于绑定的操作，如果是解除绑定列，方法如下。

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训

② 使用T-SQL语句解除绑定关系

语法格式：

sp_unbindefault [@objname=]’ object_name’

[,[@futureonly=]’ futureonly_flag’]

说明：参数object_name为要解除默认值对象绑定关系的字段名（格式为：表名.字段名）或用户定义数据类型名。用户定义数据类型与默认值对象的绑定关系解除后，所有属于该类型的列也同时解除默认值绑定；参数futureonly_flag仅用于解除用户定义数据类型默认值的绑定。当参数futureonly_flag为futureonly时，现有的属于该数据类型的列默认值不变。

（4）删除默认值对象

解除默认值对象与用户定义数据类型及表字段的绑定关系后，即可删除默认值对象。

①使用SQL Server Management Studio删除默认值对象

启动SQL Server Management Studio，展开数据库，展开【可编程性】|【默认值】，选中要删除的默认值对象名。

单击右键，选择【删除】，进入【删除对象】窗口，单击【确定】按钮即可。

② 使用T-SQL语句删除默认值对象

语法格式：

DROP DEFAULT {default}[,…n]

说明：参数default为现有默认值对象名；参数n表示可以指定多个默认值对象同时删除。

DROP DEFAULT语句不适用于DEFAULT约束。

【例8-18】解除默认值对象xb_default与用户定义数据类型sex的绑定关系，然后删除名为xb_default的默认值对象。

任务1-5 完成综合任务

1. 使用CREATE语句创建表book（书号char(6)、书名char(20)、类型char(20)、价格int）字段。给“价格”字段定义一个名为max_price的check约束 ，使得价格不超过200。

```
USE XS
CREATE table book
(
书号 char(6),
类型 char(20),
价格 int CONSTRAINT max_price CHECK( 价格 <=200)
)
GO
```

2. 定义一个名为type_rule的规则，并绑定到book表的“类型”字段，使得“类型”字段只接受该规则中列出的值（计算机类、科普类、文学类、自然科学）。

```
USE XS
GO                          --此处加GO是因为CREATE RULE必须是批查询中的第一条语句
CREATE RULE type_rule AS @类型 IN ('计算机','科普类','文学类','自然科学')
GO
USE XS
EXEC sp_bindrule 'type_rule','book.类型'
GO
```

3. 解除第2题中的规则绑定关系。

```
USE XS
EXEC sp_unbindrule 'book.类型'
GO
```

4. 删除第2题中的规则。

```
USE XS
DROP RULE type_rule
GO
```

5. 修改book表,为“类型”字段设置一个默认值约束，约束名为BookType，默认值为“NEW BOOK”。

```
USE XS
ALTER TABLE book
ADD CONSTRAINT BookType DEFAULT 'NEW BOOK' FOR 类型
GO
```

6. 从book表中删除默认值约束BookType。

```
USE XS
ALTER TABLE book
DROP CONSTRAINT BookType
GO
```

7. 定义默认值对象Day，取值为getdate()。

```
USE XS
GO
CREATE DEFAULT Day AS 'getdate()'          --该语句必须是批查询中的第一条语句
GO
```

8. 修改XSDA表，向学生表中添加字段“入学时间”。

```
USE XS
GO
ALTER TABLE XSDA
ADD 入学时间 datetime null
GO
```

9. 将默认值对象Day绑定到XSDA表的“入学时间”字段。

```
USE XS
GO
EXEC sp_bindefault 'Day','XSDA.入学时间'
GO
```

10.定义一个用户定义数据类型：Today。

```
USE XS
GO
EXEC sp_addtype Today,'datetime','NULL'
GO
```

11.将默认值对象Day绑定到用户自定义数据类型ToDay。

```
USE XS
GO
EXEC sp_bindefault 'Day','Today'
GO
```

12. 修改book表，向book表中添加字段“购买日期”，定义类型为ToDay。

```
ALTER TABLE book
ADD 购书时间 Today null
GO
```

13. 解除默认值对象Day与XSDA表“入学时间”、自定义数据类型Today的绑定关系。

```
EXEC sp_unbindefault 'XSDA.入学时间'
EXEC sp_unbindefault 'Today'
Go
```

14. 删除默认值对象Day。

```
DROP DEFAULT Day
```

15. 阐述数据完整性的作用、分类，以及每类完整性都由哪些约束实现。

数据完整性就是用于保证数据库中的数据在逻辑上的一致性、正确性和可靠性。

强制数据完整性可确保数据库中的数据质量。数据完整性一般包括3种类型：域完整性、实体完整性、参照完整性。

实现域完整性可以通过定义CHECK约束、规则、默认值约束、默认值对象等。

实体完整性可以通过索引、PRIMARY KEY约束、UNIQUE约束或IDENTITY属性来实现。

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训

参照完整性是对两个相关联的表（主表与从表）进行数据插入和删除时，保证它们之间数据的一致性。可以使用从表FOREIGN KEY定义从表的外键，主表的主键或唯一键PRIMARY KEY或UNIQUE约束（不允许为空），可实现主表与从表之间的参照完整性。定义表间参照关系应先定义主表主键约束（或唯一键约束），再对从表定义外键约束。

任务2 实现实体完整性

【任务目标】

学会使用索引、PRIMARY KEY主键约束来保证行数据完整性（即实体完整性）

学会使用UNIQUE唯一约束或IDENTITY属性来保证行数据完整性

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训

【任务描述】

按需求在XS数据库完成以下实体完整性相关的操作。

1．通过修改XSDA表，在学号(如果该字段上已经有约束，那么先把约束删除)字段上创建PRIMARY KEY约束。

2．在book表的“书名”字段上创建UNIQUE约束。

【任务分析】

练习SSMS和T-SQL两种方法实体完整性的实现，包括：UNIQUE约束、PRIMARY KEY约束的使用。

任务2-1 PRIMARY KEY约束

PRIMARY KEY约束可以在表中定义一个主键，来唯一地标识表中的行。主键可以是一列或列组合，PRIMARY KEY约束中的列不能取空值和重复值，如果PRIMARY KEY约束是由多列组合定义的，则某一列的值可以重复，但PRIMARY KEY约束定义中所有列的组合值必须唯一。一个表只能有一个PRIMARY KEY约束，而且每个表都应有一个主键。

由于PRIMARY KEY约束能确保数据的唯一，所以经常用来定义标识列。当为表定义PRIMARY KEY约束时，SQL SERVER 2008为主键列创建唯一索引，实现数据的唯一性。在查询中使用主键时，该索引可用来对数据进行快速访问。

如果已有PRIMARY KEY约束，则可对其进行修改或删除。但要修改PRIMARY KEY约束，必须先删除现有的PRIMARY KEY约束，然后再用新定义重新创建。

当向表中的现有列添加PRIMARY KEY约束时，SQL SERVER 2008检查列中现有的数据以确保现有数据遵从主键的规则（无空值和重复值）。如果PRIMARY KEY约束添加到具有空值或重复值的列上，SQL SERVER将不执行该操作并返回错误信息。

当PRIMARY KEY约束由另一表的FOREIGN KEY约束引用时，不能删除被引用的PRIMARY KEY约束，要删除它，必须先删除引用的FOREIGN KEY约束。

另外，image，text数据类型的字段不能设置为主键。

当用户在表中创建主键约束或唯一约束时，SQL Server将自动在建立这些约束的列上创建唯一索引。当用户从该表中删除主键约束或唯一约束时，创建在这些列上的唯一索引也会被自动删除。

1. 使用SQL Server Management Studio定义和删除PRIMARY KEY约束

下面介绍使用SQL Server Management Studio定义和删除PRIMARY KEY约束的方法。

如果要对XSDA表按学号字段建立PRIMARY KEY约束，方法如下。

打开XSDA表的表设计器，选定“学号”对应的这一行，在该行上单击鼠标右键，在打开的快捷菜单中选择【设置主键】，这时选定行的左边则显示一个黄色的钥匙符号，表示已经设为主键。如图8-10所示。取消主键与设置主键的方法相同，这时在快捷菜单的相同位置上出现的是【移除主键】，选择【移除主键】即可。

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训

如果主键由多列组成，可以先选中这些列，然后再设置主键图标。选择多列的方法是：按住ctrl键再单击相应的列，如果列是连续的也可以按住shift键。

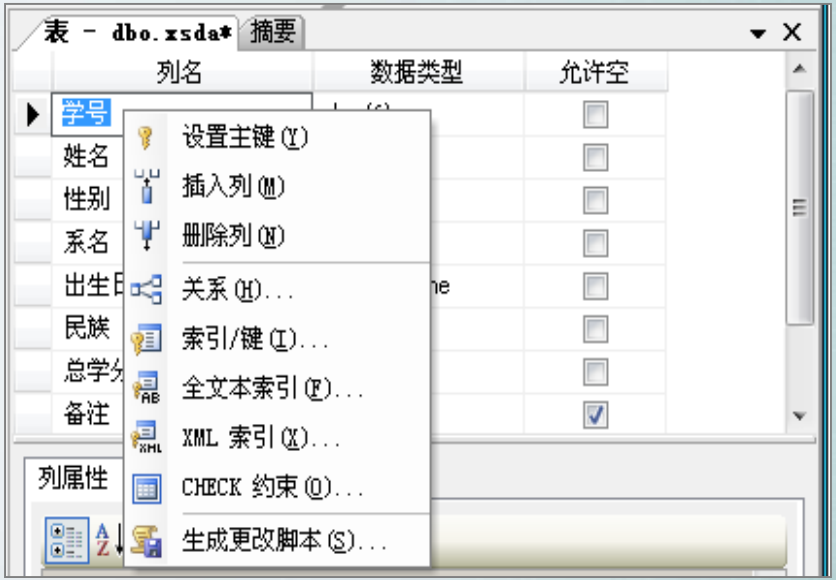


图8-10 为XSDA表的学号列设置主键

2. 使用T-SQL语句定义和删除PRIMARY KEY约束

可以使用T-SQL语句定义和删除PRIMARY KEY约束。

（1）创建表时定义PRIMARY KEY约束

语法格式：

```
CREATE TABLE table_name                          /*指定表名
(column_name datatype NOT NULL| NULL             /*定义列名、数据类型、是否空值
[CONSTRAINT constraint_name]                     /*指定约束名
PRIMARY KEY                                     /*定义约束类型
[ CLUSTERED | NONCLUSTERED]                     /*定义约束的索引类型
```

【例8-19】创建XSDA2表时对学号字段创建PRIMARY KEY约束。

项目8 实现数据完整性


```
use xs
go
create table xsda2
(
学号 char(6) not null constraint xh_pk primary key,
姓名 char(8)not null,
性别 char(2) not null,
系名 char(10) not null,
出生日期 smalldatetime not null,
民族 char(4) not null
)
```

（2）修改表时定义PRIMARY KEY约束

语法格式：

```
ALTER TABLE table_name
ADD [CONSTRAINT constraint_name] PRIMARY KEY
[ CLUSTERED | NONCLUSTERED]
（column）
```

说明：ADD CONSTRAINT关键字用于说明对table_name表增加一个约束，约束名由constraint_name指定，约束类型为PRIMARY KEY；索引字段由column参数指定。创建的索引类型由关键字CLUSTERED，NONCLUSTERED指定。

【例8-20】假设已经在XS数据库中创建了KCXX表，然后通过修改表，对课程编号字段创建PRIMARY KEY约束。

```
use xs
go
alter table kcxx
add constraint kcbh_pk primary key (课程编号)
```

也可以用如下方法

```
use xs
go
alter table kcxx
add  primary key (课程编号)
```

3. 删除PRIMARY KEY约束

语法格式：

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_name[,…n]
```

【例8-21】删除上例中创建的PRIMARY KEY约束kcbh_pk。

```
alter table kcxx
drop  constraint kcbh_pk
```

任务2-2 UNIQUE约束

如果要确保一个表中的非主键列不输入重复值，应在该列上定义UNIQUE约束（唯一约束）。在允许空值的列上保证唯一性时，应使用UNIQUE约束而不是PRIMARY KEY约束，不过在该列中只允许有一个NULL值。FOREIGN KEY约束也可引用UNIQUE约束。

PRIMARY KEY约束与UNIQUE约束的主要区别如下：

（1）一个数据表只能定义一个PRIMARY KEY约束，但一个表中可根据需要对不同的列定义若干个UNIQUE约束。

（2）PRIMARY KEY字段的值不允许为NULL，而UNIQUE字段的值可取NULL。

（3）一般创建PRIMARY KEY约束时，系统会自动产生索引，索引的默认类型为簇索引（clustered）。创建UNIQUE约束时，系统会自动产生一个UNIQUE索引，索引的默认类型为非簇索引(nonclustered)。

PRIMARY KEY约束与UNIQUE约束的相同点在于：二者均不允许表中对应字段存在重复值。

1. 使用SQL Server Management Studio定义和删除UNIQUE约束

假设已经为XSDA表增加了一个字段：身份证号码 char(18)。并且已经为表中原有的记录输入了不同的身份证号码。要求对“身份证号码”列创建UNIQUE约束，以保证该列取值的唯一性，方法如下：

（1）启动SQL Server Management Studio，打开XSDA表的表设计器，在表设计器中右击，出现如图8-10所示的快捷菜单。

（2）在快捷菜单中选择【索引/键】选项，出现“属性”窗口，在此窗口中单击【添加】按钮，在【类型】下拉列表中选择“唯一键”，在【列】下拉列表中选择“身份证号码”，如图8-11所示，然后单击【关闭】按钮，则完成了指定列的唯一性约束设置。

若要删除刚才创建的UNIQUE约束，只需进入如图8-11所示的“索引/键”窗口中，在【选定的主/唯一键或索引】下拉框中选择要删除的UNIQUE约束的索引名，再单击【删除】按钮，即删除了指定的UNIQUE约束。

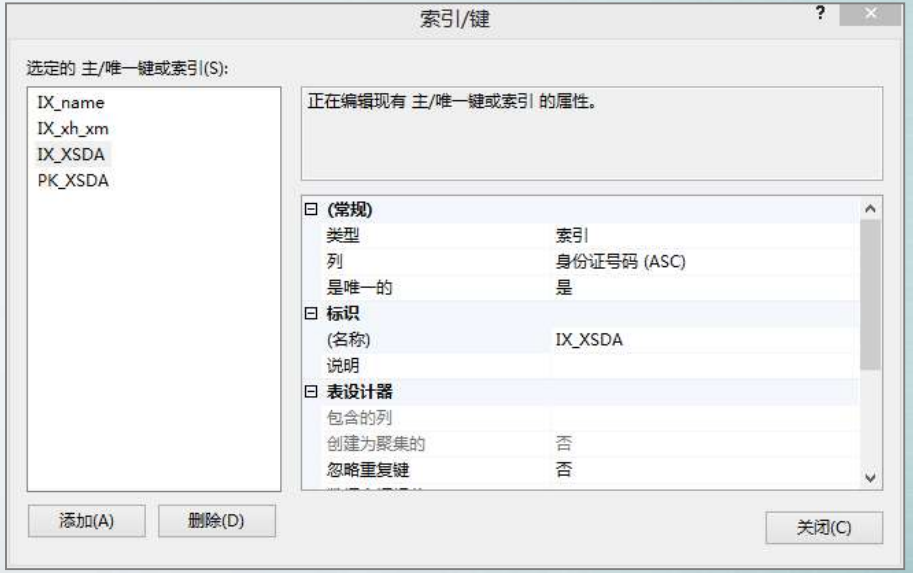


图8-11 在“索引/键”窗口中设置UNIQUE约束

2. 使用T-SQL语句定义和删除UNIQUE约束

可以使用T-SQL语句定义和删除UNIQUE约束。

（1）创建表时定义UNIQUE约束

语法格式：

```
CREATE TABLE table_name                          /*指定表名
(column_name datatype NOT NULL| NULL             /*定义列名、数据类型、是否空值
[CONSTRAINT constraint_name]                     /*约束名
UNIQUE                                           /*定义约束类型
[ CLUSTERED | NONCLUSTERED]                      /*定义约束的索引类型
[,…n])                                           /*n表示可定义多个字段
```

【例8-22】在创建XSDA3表时对身份证号码字段定义UNIQUE约束。

```
create table xsda3
( 学号 char(6) not null constraint xh1_pk primary key,
 姓名 char(8) not null,
 性别 char(2) not null,
身份证号码 char(18) constraint sfzhm_uk unique,
系名 char(10) not null,
出生日期 smalldatetime not null,
民族 char(4) not null
)
```

（2）修改表时定义UNIQUE约束

语法格式：

```
ALTER TABLE table_name
ADD [CONSTRAINT constraint_name] UNIQUE
[ CLUSTERED | NONCLUSTERED]
（column[,…n]）
```

说明：各项参数的说明同上述的PRIMARY KEY约束。

项目8 实现数据完整性

【例8-23】给KCXX表中的课程编号字段定义UNIQUE约束。

```
use xs
go
alter table kcxx
add constraint kcbh_uk unique (课程编号)
go
```

（3）删除UNIQUE约束

语法格式：

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_name[,…n]
```

【例8-24】删除前面例中创建的UNIQUE约束。

```
use xs
alter table kcxx
drop constraint kcbh_uk
```

任务2-3 完成综合任务

1．通过修改XSDA表，在学号(如果该字段上已经有约束，那么先把约束删除)字段上创建PRIMARY KEY约束。

```
ALTER TABLE XSDA
ADD CONSTRAINT XSDA_pk PRIMARY KEY CLUSTERED （学号）
GO
--删除主键索引要与建立主键索引的方法匹配，如果用SSMS建立只能在SSMS中删除主键；--如果是用T-SQL建立的主键约束，则可以用T-SQL和SSMS两种方法删除主键约束
ALTER TABLE XSDA
DROP CONSTRAINT XSDA_pk
GO
```

2. 在book表的“书名”字段上创建UNIQUE约束。

```
USE XS
ALTER TABLE book
ADD CONSTRAINT book_uk UNIQUE NONCLUSTERED (书号)
GO
```

任务3 实现参照完整性

【任务目标】

- 学会使用从表的FOREIGN KEY外键约束与主表的定义主键PRIMARY KEY或唯一键UNIQUE约束（不允许为空）实现主表与从表之间的参照完整性

【任务描述】

- 设“KCXX”表为主表，“XSCJ”表为从表，通过修改表定义两个表之间的参照关系。

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训

【任务分析】

练习参照完整性的实现。设置参照完整性的意义简单地说就是控制数据一致性，尤其是不同表之间关系的规则。“参照完整性生成器”可以帮助我们建立规则，控制记录如何在相关表中被插入、更新或删除，这些规则将被写到相应的表触发器中。比如，主表中没有“张三”，那么你就不能在子表中给张三这个人添加相关的内容；如果在主表中将张三删了，子表中和张三有关的内容也会被删掉。

任务3-1 FOREIGN KEY（外键）

对两个相关联的表（主表与从表）进行数据插入和删除时，通过参照完整性保证它们之间数据的一致性。

利用FOREIGN KEY定义从表的外键，利用PRIMARY KEY或UNIQUE约束定义主表的主键或唯一键（不允许为空），可实现主表与从表之间的参照完整性。

定义表间参照关系：先定义主表主键约束（或唯一键约束），再对从表定义外键约束。

1. 使用SQL Server Management Studio定义表间的参照关系

例如要建立XSDA表与XSCJ表之间的参照关系，方法如下：

（1）首先定义主表的主键，在此定义XSDA表中学号字段为主键。

（2）选择SQL Server Management Studio目录树中XS数据库目录下的【数据库关系图】图标右击，出现如图8-12所示的快捷菜单。

（3）在快捷菜单中选择【新建数据库关系图】选项，进入如图8-13所示的“添加表”界面，从可用表中选择要添加到关系图中的表，本例中选择了XSDA表与XSCJ表。

（4）单击【添加】按钮，然后单击【关闭】，进入如图8-14所示的关系图界面。

（5）在关系图上，将鼠标指向主表的主键并拖动到从表。对于本例：将XSDA表中的学号字段拖动到从表XSCJ，出现如图8-15所示的“表和列”界面，在此界面中，选择主表中的主键及从表中的外键，然后单击【确定】按钮，进入如图8-16所示的“外键关系”界面。

（6）退出如图8-16所示的界面，并根据提示，将关系图的有关信息存盘，即创建了主表与从表之间的参照关系。

项目8 实现数据完整性

任务1 任务2 任务3 实训

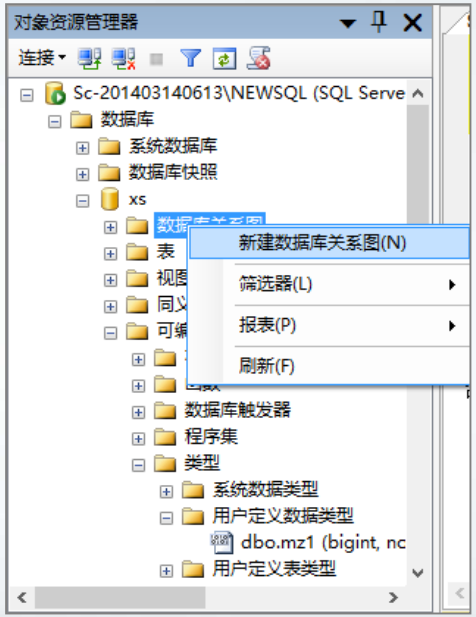


图8-12 关系图的快捷菜单

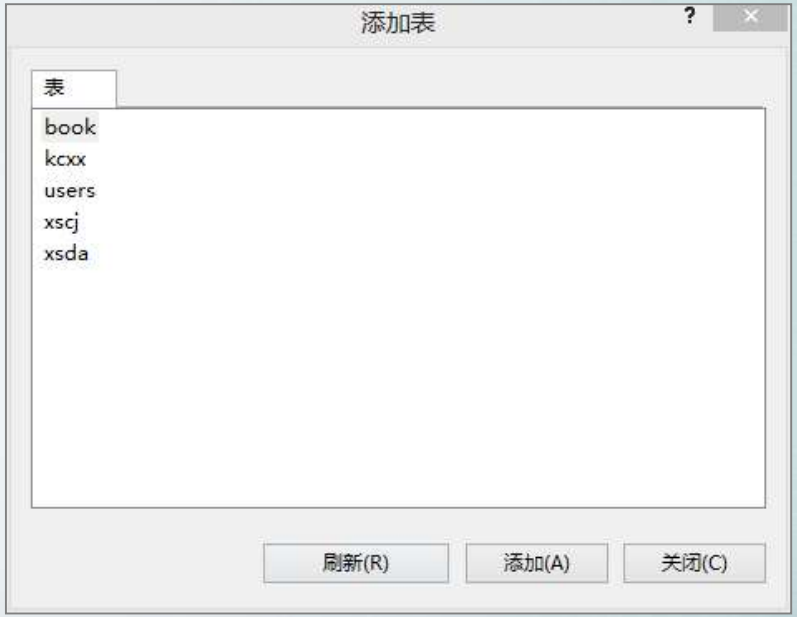


图8-13 “添加表”的界面

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训

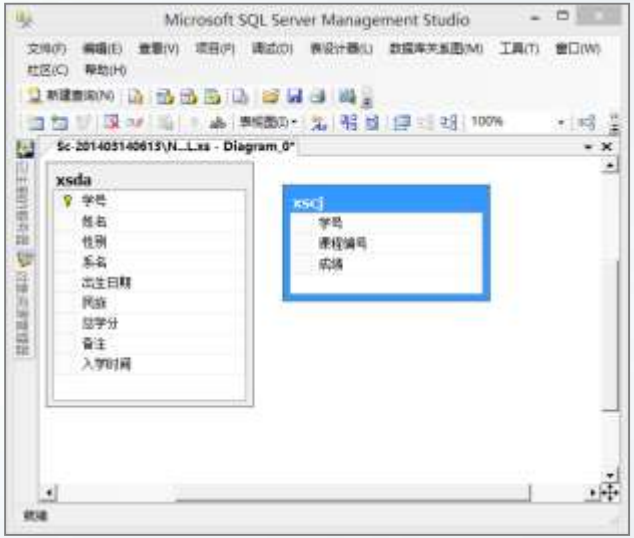


图8-14 关系图界面

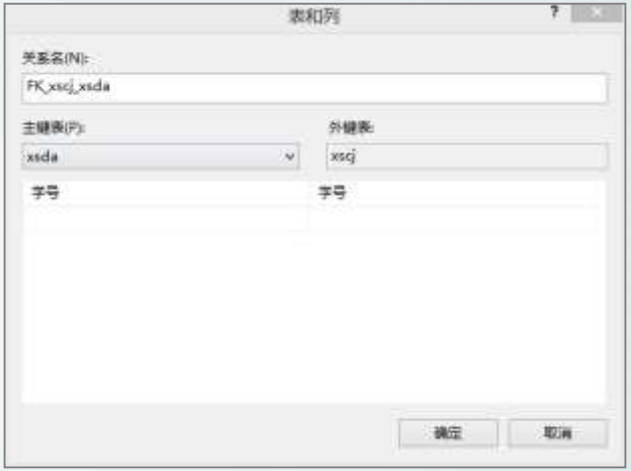


图8-15 “表和列”界面

图8-16 “外键关系”界面

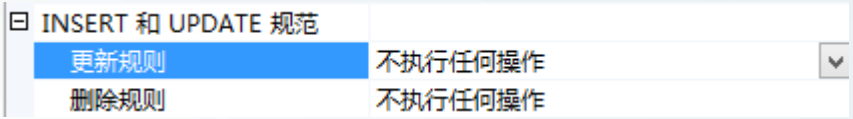
读者可在主表和从表中插入或删除相关数据，验证它们之间的参照关系。

另外，在SQL Server Management Studio 中，读者也可以在主表或从表的表设计器中右击鼠标，在快捷菜单中选择“关系”选项，单击“添加”按钮，然后通过单击“表和列规范”右侧的按钮，在弹出的【表和列】对话框中定义主表和从表之间的参照关系。

为了提高查询效率，在定义主表与从表间的参照关系前，可考虑先对从表的外键定义索引，然后再定义主表与从表间的参照关系。

项目8 实现数据完整性

任务1 | 任务2 | 任务3 | 实训



建立参照关系之后：

(1)如果要删除数据，必须先删除从表的数据，再删除主表数据。（不执行任何操作时）

如果删除数据，那么删除主表数据，从表数据也会执行相应删除操作。（级联）

（2）如果要更新数据，更新主表的数据，会同时更新从表数据。（级联）

如果要更新数据，不管是先更新哪一个表，主表、从表的数据无法更新。（不执行任何操作时）

（3）如果要插入数据，先从主表插入数据，再从从表进行插入。（级联、不执行任何操作时）

2. 使用SQL Server Management Studio删除表间的参照关系

如果要删除前面建立的XSDA表与XSCJ表之间的参照关系，可按照以下步骤进行：

（1）打开XSDA表的表设计器，右击，出现一个快捷菜单，如图8-17所示。

（2）在快捷菜单中选择【关系】选项，出现如图8-16所示的“外键关系”界面。

（3）在“外键关系”界面的【选定的关系】中选择要删除的关系，然后单击【删除】按钮，再单击【关闭】按钮。即可删除参照关系。

项目8 实现数据完整性

任务1 任务2 任务3 实训

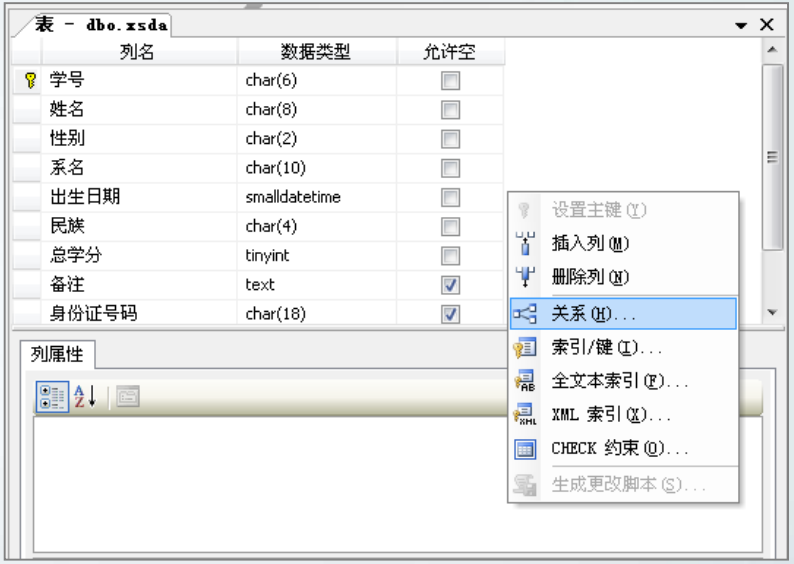


图8-17 表设计器的快捷菜单

3. 使用T-SQL语句定义表间的参照关系

定义表间参照关系，需要先定义主表主键（或唯一键），再对从表定义外键约束。

前面已经介绍了定义主键（PRIMARY KEY约束）及唯一键（UNIQUE约束）的方法，在此将介绍通过SQL命令定义外键的方法。

（1）创建表时定义外键约束

语法格式：

```
CREATE TABLE table_name                                    /*指定表名
(column_name  datatype  [CONSTRAINT constraint_name] [FOREIGN KEY]
REFERENCES ref_table (ref_column) [ON DELETE CASCADE|ON UPDATE CASCADE]
)
```

说明：参数table_name为所建从表的表名；column_name为定义的字段名；字段类型由参数datatype指定；关键字FOREIGN KEY指明该字段为外键，并且该外键与参数ref_table指定的主表中的主键对应，主表中主键字段由参数ref_column指定。ON DELETE CASCADE表示级联删除，当在主键表中删除外键引用的主键记录时，为防止孤立外键的产生，将同时删除外键表中引用它的外键记录；ON UPDATE CASCADE表示级联更新，当在主键表中更新外键引用的主键记录时，外键表中引用它的外键记录也一起被更新。

【例8-25】在XS数据库中创建主表XSDA4，定义XSDA4.学号为主键，然后创建从表XSCJ4，定义XSCJ4.学号为外键。

```
create table xsda4
( 学号 char(6) not null constraint xh4_pk primary key,
  姓名 char(8) not null ,
  性别 char(2) not null,
  系名 char(10) not null,
  出生日期 smalldatetime not null,
  民族 char(4) not null,
  总学分 tinyint null,
  备注 text null)
--定义外键
create table xscj4
(  学号 char(6) not null foreign key references xsda4(学号) on update cascade,
  课程编号 char(3) not null,
  成绩 tinyint)
```

（2）修改表时定义外键约束

语法格式：

```
ALTER TABLE table_name
ADD [CONSTRAINT constraint_name]
FOREIGN KEY (column [,…n])
REFERENCES ref_table (ref_column[,…n])
[ON DELETE CASCADE|ON UPDATE CASCADE]
```

说明：参数table_name指定被修改的从表名；column为从表中外键的列名，当外键由多列组合时列之间用逗号分隔；ref_table为主表的表名；ref_column为主表中主键的列名，当主键由多列组合时，列名之间用逗号分隔；n表示可指定多列。

项目8 实现数据完整性

【例8-26】 假设XS数据库中KCXX表为主表，KCXX.课程编号字段已经定义为主键，XSCJ表为从表，要求将XSCJ.课程编号字段定义为外键。

```
use xs
  alter table xscj
  add constraint kc_foreign foreign key(课程编号)
  references kcxx(课程编号)
```

4. 使用T-SQL语句删除表间的参照关系

删除表间的参照关系，实际上删除从表的外键约束即可。语法格式与前面其他约束删除的格式相同。

【例8-27】删除上例对XSCJ. 课程编号字段定义的外键约束。

```
alter table xscj
drop constraint kc_foreign
```

任务3-2 完成综合任务

设“KCXX”表为主表，“XSCJ”表为从表，通过修改表定义两个表之间的参照关系。

首先定义KCXX表中学号字段为主键->【数据库关系图】右击->【新建数据库关系图】->“添加表”->选择了KCXX表与XSCJ表->【添加】->【关闭】->向主表KCXX的主键拖动到从表->【确定】->退出->存盘，即创建了主表与从表之间的参照关系。

实训八 实现sale数据库完整性

1．根据你的理解，简述sale数据库需要设置哪些主键，写出SQL语句。

2．在开发时需要保证ProPut表与Product表之间的参照完整性，即向PrOut表录入或修改产品编号ProNo时，它必须在Product表中存在。

3．根据你的理解，简述sale数据库还需要设置哪些外键，写出SQL语句。

4．在销售表ProOut上对数据Quantity列的值进行限制，使其值>=1时有效。

5．在销售表ProOut上对SaleDate列进行设定，当不输入其值时，使系统默认其值为当前日期。

小结

本项目主要介绍了数据完整性技术，内容包括：数据完整性的概念、分类及域完整性、实体完整性、参照完整性的实现。需要掌握的主要内容如下：

1．数据完整性的概念：数据完整性就是用于保证数据库中的数据在逻辑上的一致性、正确性和可靠性。强制数据完整性可确保数据库中的数据质量。

2．数据完整性一般包括3种类型：域完整性、实体完整性、参照完整性。

3．域完整性的实现：域完整性是指给定列输入的有效性，即保证指定列输入的数据具有正确的数据类型、格式和有效的数据范围。实现域完整性可通过定义相应的CHECK约束、默认值约束、默认值对象、规则对象等方法来实现。

（1） CHECK约束：是字段输入内容的验证规则。通过ALTER TABLE 或CREATE TABLE的CHECK关键字创建，是对列中的值进行限制的首选标准方法（可以对一列或多列定义多个约束）。

（2）规则：规则是一种数据库对象，可以绑定到一列或多个列上，还可以绑定到用户定义数据类型上，规则定义之后可以反复使用。列或用户定义数据类型只能有一个绑定的规则。但是，列可以同时具有规则和多个CHECK约束。

规则作为独立的数据库对象，使用它要首先定义，然后绑定到列或用户定义数据类型；不需要时要先解除绑定，然后删除规则。

（3）默认值约束：在创建表或修改表时，可以定义默认值约束。

（4）默认值对象：先定义默认值对象，然后将该默认值对象绑定到表的相应字段上或用户定义数据类型上。不需要时要先解除绑定，然后删除默认值对象。

4．实体完整性的实现：实体完整性是用于保证数据表中每一个特定实体的记录都是唯一的。通过索引、UNIQUE约束、PRIMARY KEY约束或IDENTITY属性可以实现数据的实体完整性。

（1）PRIMARY KEY约束：PRIMARY KEY约束可以在表中定义一个主键，来唯一地标识表中的行。主键可以是一列或列组合，PRIMARY KEY约束中的列不能取空值和重复值。一个表只能有一个PRIMARY KEY约束，而且每个表都应有一个主键。

（2）UNIQUE约束：UNIQUE约束可以确保一个表中的非主键列不输入重复值，在允许空值的列上保证唯一性时，应使用UNIQUE约束而不是PRIMARY KEY约束。

5．参照完整性的实现：对两个相关联的表（主表与从表）进行数据更新和删除时，通过参照完整性保证它们之间数据的一致性。

先利用PRIMARY KEY或UNIQUE约束定义主表的主键或唯一键（不允许为空），再利用FOREIGN KEY定义从表的外键，可实现主表与从表之间的参照完整性。