第9章 数据完整性

主要内容

返回总目录

- 数据完整性概述
- 规　　则
- 缺　省　值

数据库中的数据是从外界输入的，而数据的输入由于种种原因，会发生输入无效或错误信息。保证输入的数据符合规定，成为了数据库系统，尤其是多用户的关系数据库系统首要关注的问题。数据完整性因此而提出。本章将讲述数据完整性的概念及其在 SQL Server 中的实现方法。

9.1 数据完整性概述

数据完整性（Data Integrity）是指数据的精确性（Accuracy）和可靠性（Reliability）。它是应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的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）。

9.1.1 实体完整性（Entity Integrity）

实体完整性规定表的每一行在表中是惟一的实体。表中定义的 UNIQUE、PRIMARY KEY 和 IDENTITY 约束就是实体完整性的体现。

9.1.2 域完整性（Domain Integrity）

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。

9.1.3 参照完整性（Referential Integrity）

参照完整性是指两个表的主关键字和外关键字的数据应对应一致。它确保了有主关键字的表中对应其它表的外关键字的行存在，即保证了表之间的数据的一致性，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关键字和主关键字之间或外关键字和惟一性关键字之间的关系上的。在 SQL Server 中，参照完整性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关键字值改变;
- 禁止删除在从表中的有对应记录的主表记录。

9.1.4 用户定义的完整性（User-defined Integrity）

不同的关系数据库系统根据其应用环境的不同，往往还需要一些特殊的约束条件。用户定义的完整性即是针对某个特定关系数据库的约束条件，它反映某一具体应用所涉及的数据必须满足的语义要求。SQL Server 提供了定义和检验这类完整性的机制，以便用统一的系统方法来处理它们，而不是用应用程序来承担这一功能。其它的完整性类型都支持用户定义的完整性。

SQL Server 提供了一些工具来帮助用户实现数据完整性，其中最主要的是：规则（Rule）、缺省值（Default）、约束（Constraint）和触发器（Trigger）。其中约束在前

面的章节中已经介绍，触发器将在后面的章节中介绍。本章将介绍规则和缺省值。

9.2 规则

规则（Rule）就是数据库中对存储在表的列或用户自定义数据类型中的值的规定和限制。规则是单独存储的独立的数据库对象。规则与其作用的表或用户自定义数据类型是相互独立的，即表或用户自定义对象的删除、修改不会对与之相连的规则产生影响。规则和约束可以同时使用，表的列可以有一个规则及多个 CHECK 约束。规则与 CHECK 约束很相似，相比之下，使用在 ALTERTABLE 或 CREATE TABLE 命令中的 CHECK 约束是更标准的限制列值的方法，但 CHECK 约束不能直接作用于用户自定义数据类型。

9.2.1 创建规则

1．用 CREATE RULE 命令创建规则

CREATE RULE 命令用于在当前数据库中创建规则，其语法如下：

CREATE RULE rule_name AS condition_expression

其中 condition_expression 子句是规则的定义。condition_expression 子句可以是能用于 WHERE 条件子句中的任何表达式，它可以包含算术运算符、关系运算符和谓词（如 IN、LIKE、BETWEEN 等）。

> ! condition_expression 子句中的表达式必须以字符“@”开头。

例 9-1：创建雇佣日期规则 hire_date_rule。

```
create rule hire_date_rule
as @hire_date >= '1980-01-01' and @hire_date <= getdate()
```

例 9-2：创建工作级别规则 job_level_rule。

```
create rule job_level_rule
as @job_level in ('1','2','3','4','5')
```

例 9-3：创建评分规则 grade_rule。

```
create rule grade_rule
as @value between 1 and 100
```

例 9-4：创建字符规则 my_character_rule1。

```
create rule my_character_rule1
as @value like '[a-z]%[0-9]'    /* 字符串必须以‘a’到‘f’的字母开头，以’0’到’9’的数字结尾 */
```

2．用 Enterprise Manager 创建规则

在 Enterprise Manager 中选择数据库对象“Rules”，单击右键，从快捷菜单中选择“New Rule”选项，即会弹出如图 9-1 所示的创建规则属性对话框。输入规则名称和表达式之后，单击“确定”按钮，即完成规则的创建。

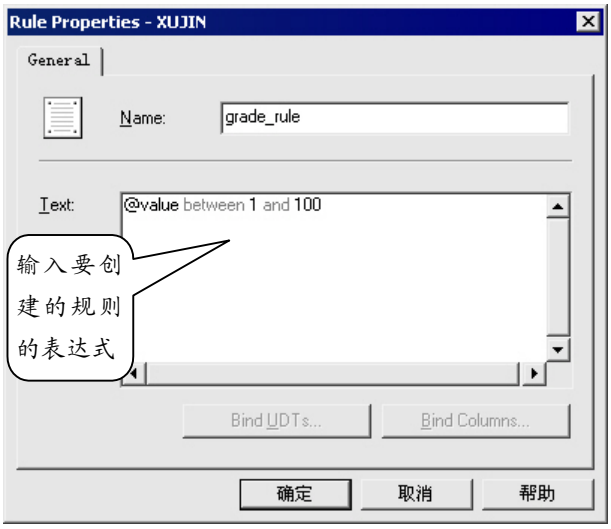


图9-1 创建规则属性对话框

9.2.2 查看规则

1．用 Enterprise Manager 查看规则

在 Enterprise Manager 中选择“Rules”对象，即可从右边的任务板中看到规则的大部分信息，包括规则的名称、所有者、创建时间等，如图 9-2 所示。在 SQL Server 2000 中不像 7.0 版本可以直接在任务板中看到规则的表达式，这需要查看规则的属性。可以选择要查看的规则，单击右键，从快捷菜单中选择“Properties”选项，即会出现如图 9-3 所示的对话框，可以从中编辑规则的表达式。修改规则的名称可以通过 Sp_rename 系统存储过程进行，也可以直接在图 9-2 中用右键单击要修改的规则，从快捷菜单中选择“重命名（Rename）”菜单项，进行名称修改。

2．用存储过程 Sp_helptext 查看规则

使用 Sp_helptext 存储过程可以查看规则的细节，其语法如下：

sp_helptext [@objname =] 'name'

其中[@objname =] 'name'子句指明对象的名称，用 Sp_helptext 存储过程查看的对象可以是当前数据库中的规则、缺省值、触发器、视图或未加密的存储过程。

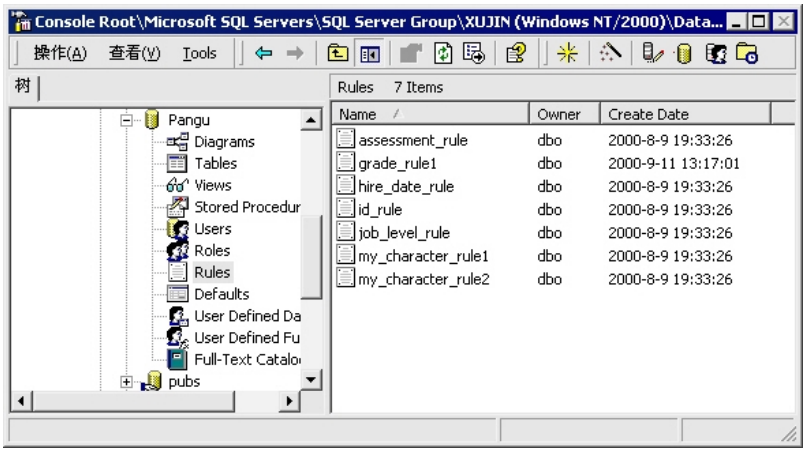


图9-2 任务板中的规则信息

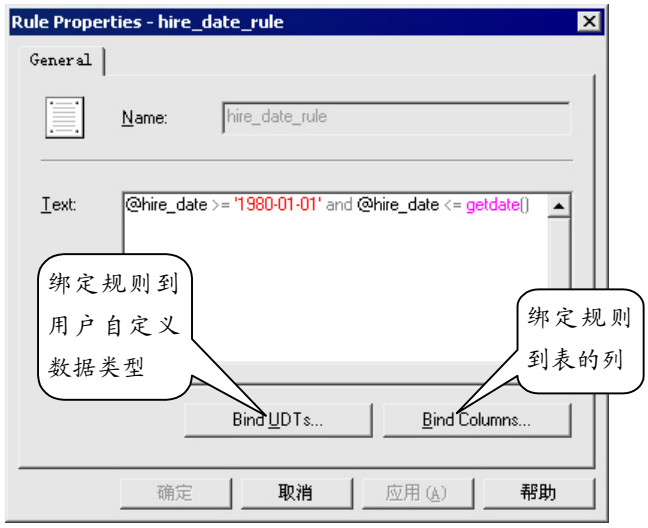


图9-3 规则属性对话框

例 9-5：查看规则 hire_date_rule。

```
exec sp_helptext hire_date_rule
```

运行结果如下：

```
Text
------------------------------------------------------------------------------
create rule hire_date_rule
as @hire_date >= '1980-01-01' and @hire_date <= getdate()
```

9.2.3 规则的绑定与松绑

创建规则后，规则仅仅只是一个存在于数据库中的对象，并未发生作用。需要将规则

与数据库表或用户自定义对象联系起来，才能达到创建规则的目的。联系的方法称为“绑定”。所谓绑定就是指定规则作用于哪个表的哪一列或哪个用户自定义数据类型。表的一列或一个用户自定义数据类型只能与一个规则相绑定，而一个规则可以绑定多对象，这正是规则的魅力所在。解除规则与对象的绑定称为“松绑”。

1．用存储过程 Sp_bindrule 绑定规则

存储过程 Sp_bindrule 可以绑定一个规则到表的一个列或一个用户自定义数据类型上。其语法如下：

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

各参数说明如下：

- [@rulename =] 'rule'

指定规则名称。

- [@objname =] 'object_name'

指定规则绑定的对象。

- 'futureonly'

此选项仅在绑定规则到用户自定义数据类型上时才可以使用。当指定此选项时，仅以后使用此用户自定义数据类型的列会应用新规则，而当前已经使用此数据类型的列则不受影响。

例 9-6：绑定规则 hire_date_rule 到用户自定义数据类型 hire_date 上。

```
exec sp_bindrule hire_date_rule, hire_date
```

运行结果如下：

```
Rule bound to data type.
The new rule has been bound to column(s) of the specified user data type.
```

例 9-7：绑定规则 hire_date_rule 到用户自定义数据类型 hire_date 上，带'futureonly'选项。

```
exec sp_bindrule hire_date_rule, hire_date, 'futureonly'
```

运行结果如下：

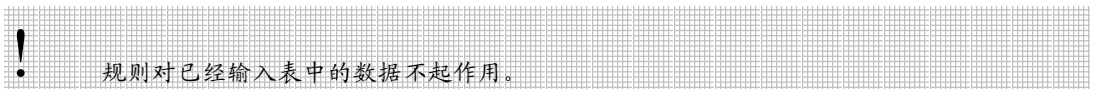
```
Rule bound to data type.
```

例 9-8：绑定规则 my_rule 到 orders 表的字段 order_id。

```
exec sp_bindrule id_rule, 'orders.[order_id]'
```

运行结果如下：

```
Rule bound to table column.
```



规则对已经输入表中的数据不起作用。

! 规则所指定的数据类型必须与所绑定的对象的数据类型一致，且规则不能绑定一个数据类型为 TEXT、MAGE 或 TIMESTAMP 的列。

! 与表的列绑定的规则优先于与用户自定义数据类型绑定的列。因此，如果表的列的数据类型与规则 A 绑定，同时列又与规则 B 绑定，则以规则 B 为列的规则。

! 你可以直接用一个新的规则来绑定列或用户自定义数据类型，而不需要先将其原来绑定的的规则解除，系统会将旧规则覆盖。

2．用存储过程 Sp_unbindrule 解除规则的绑定

存储过程 Sp_unbindrule 可解除规则与列或用户自定义数据类型的绑定，其语法如下：

sp_unbindrule [@objname =] 'object_name'
[,'futureonly']

其中'futureonly'选项同绑定时一样，仅用于用户自定义数据类型，它指定现有的用此用户自定义数据类型定义的列仍然保持与此规则的绑定。如果不指定此项，所有由此用户自定义数据类型定义的列也将随之解除与此规则的绑定。

例 9-9：解除规则 hire_date_rule 与用户自定义数据类型 birthday 的绑定，带'futureonly'选项。

```
exec sp_unbindrule birthday, 'futureonly'
```

运行结果如下：

```
 (1 row(s) affected)
Rule unbound from data type.
```

3．用 Enterprise Manager 管理规则的绑定

在 Enterprise Manager 中，选择要进行绑定设置的规则，单击右键，从快捷菜单中选择“Properties”选项，即会出现如图 9-3 所示的规则属性对话框。图中的“Bind UDTs…”按钮用于绑定用户自定义数据类型，“Bind Columns…”按钮用于绑定表的列。

在图 9-3 中单击“Bind UDTs…”按钮，则出现如图 9-4 所示的绑定规则到用户自定义数据类型的对话框；单击“Bind Columns…”按钮，则出现如图 9-5 所示的绑定规则到表的列的对话框。

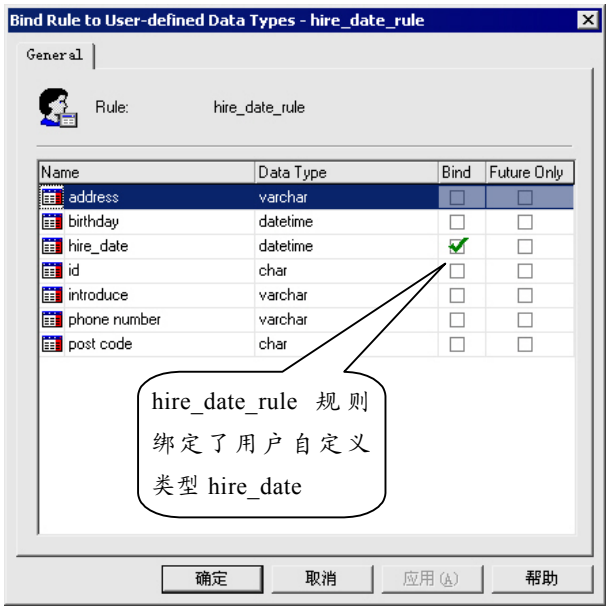


图9-4 管理规则与用户自定义数据类型之间的绑定

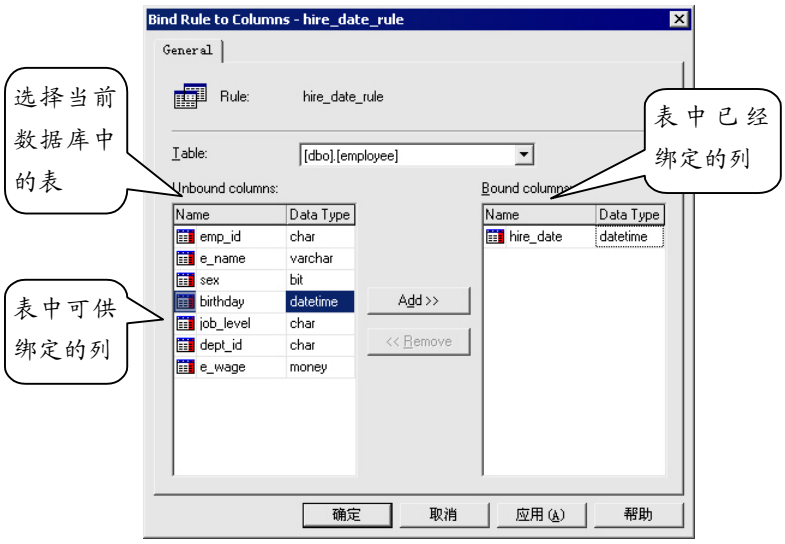


图9-5 管理规则与表的列之间的绑定

9.2.4 删除规则

可以在 Enterprise Manager 中选择规则，单击右键，从快捷菜单中选择“Delete”选项删除规则，也可使用 DROP RULE 命令删除当前数据库中的一个或多个规则。其语法如下：

DROP RULE {rule_name} [,...n]

! 在删除一个规则前，必须先将与其绑定的对象解除绑定。

例 9-10：删除多个规则

```
drop rule mytest1_rule,mytest2_rule
```

9.3 缺省值

缺省值（Default）是往用户输入记录时没有指定具体数据的列中自动插入的数据。缺省值对象与 ALTER TABLE 或 CREATE TABLE 命令操作表时用 DEFAULT 选项指定的缺省值功能相似，但缺省值对象可以用于多个列或用户自定义数据类型，它的管理与应用同规则有许多相似之处。表的一列或一个用户自定义数据类型也只能与一个缺省值相绑定。

9.3.1 创建缺省值

1．用 CREATE DEFAULT 命令创建缺省值

CREATE DEFAULT 命令用于在当前数据库中创建缺省值对象，其语法如下：

CREATE DEFAULT default_name AS constant_expression

其中 constant_expression 子句是缺省值的定义。constant_expression 子句可以是数学表达式或函数，也可以包含表的列名或其它数据库对象。

例 9-11：创建生日缺省值 birthday_defa。

```
create default birthday_defa
as '1978-1-1'
```

例 9-12：创建姓名缺省值 name_defa。

```
create default name_defa
as user
```

2．用 Enterprise Manager 创建缺省值

在 Enterprise Manager 中选择数据库对象“Defaults”，单击右键，从快捷菜单中选择“New Default”选项，即会弹出如图 9-6 所示的创建缺省值属性对话框。输入缺省值名称和值表达式之后，单击“确定”按钮，即完成缺省值的创建。

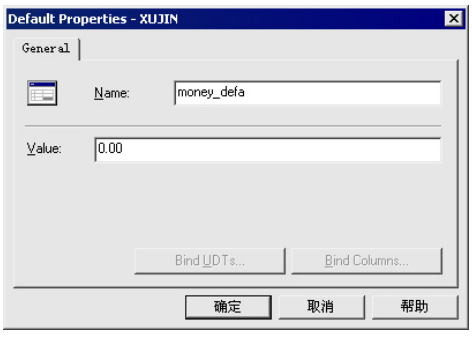


图9-6 创建缺省值属性对话框

9.3.2 查看缺省值

1．用 Enterprise Manager 查看缺省值

在 Enterprise Manager 中选择“Defaults”对象，即可从右边的任务板中看到缺省值的大部分信息。如图 9-7 所示。也可以选择要查看的缺省值，单击右键，从快捷菜单中选择“Properties”选项，即会出现如图 9-8 所示的缺省值属性对话框，可以从中编辑缺省值的值表达式。修改缺省值名称的方法与修改规则名称的方法相同，可以用 Sp_rename 存储过程修改，也可以在企业管理器的任务板窗口中直接修改。

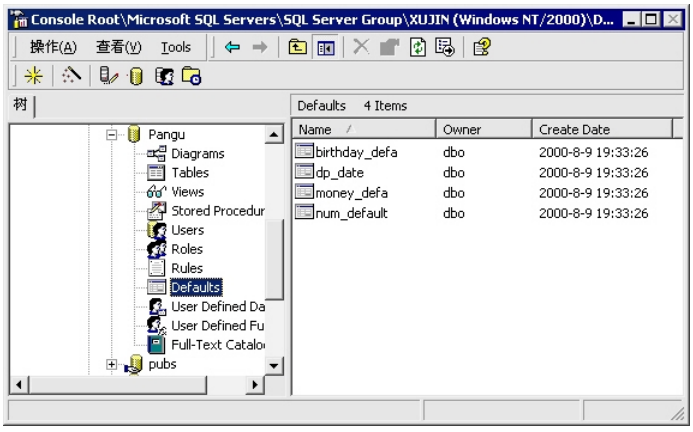


图9-7 任务板中的缺省值信息

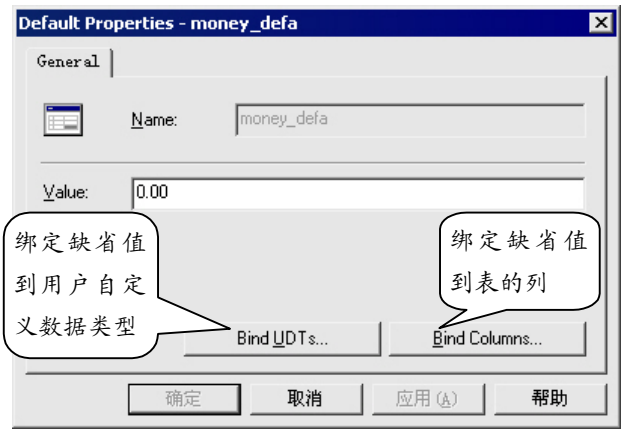


图9-8 缺省值属性对话框

2．用存储过程 Sp_helptext 查看缺省值

使用 Sp_helptext 存储过程可以查看缺省值的细节。

例 9-13：查看缺省值 today。

```
exec sp_helptext today
```

运行结果如下：

```
------------------------------------------------------------
Text
------------------------------------------------------------
create default [dp_date] as getdate()
```

9.3.3 缺省值的绑定与松绑

创建缺省值后，缺省值仅仅只是一个存在于数据库中的对象，并未发生作用。同规则一样，需要将缺省值与数据库表或用户自定义对象绑定。

1．用 Enterprise Manager 管理缺省值的绑定

在 Enterprise Manager 中，选择要进行绑定设置的缺省值，单击右键，从快捷菜单中选择“Properties”选项，即会出现如图 9-8 所示的缺省值属性对话框。

图 9-8 中的“Bind UDTs…”按钮用于绑定用户自定义数据类型，“Bind Columns…”按钮用于绑定表的列。在图 9-8 中单击“Bind UDTs…”按钮，则出现如图 9-9 所示的绑定缺省值到用户自定义数据类型的对话框；单击“Bind Columns…”按钮，则出现如图 9-10 所示的绑定缺省值到表的列的对话框。用它们来管理缺省值与表的列以及用户自定义数据类型之间的绑定非常方便。

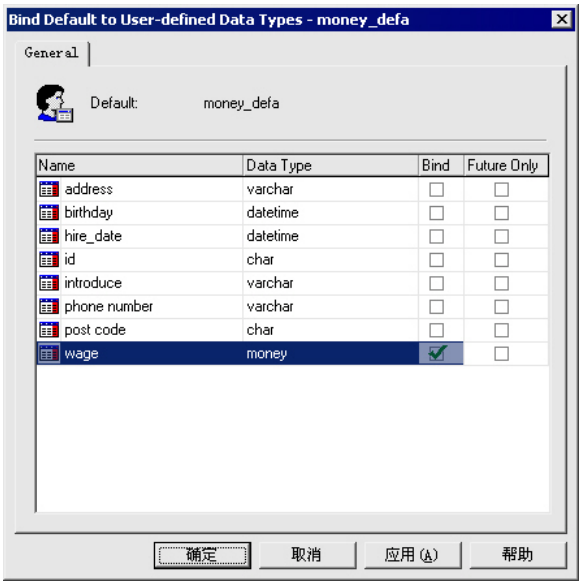


图9-9 管理缺省值与用户自定义数据类型之间的绑定

图9-10 管理缺省值与表的列之间的绑定

2．用存储过程 Sp_bindefault 绑定缺省值

存储过程 Sp_bindefault 可以绑定一个缺省值到表的一个列或一个用户自定义数据类型上。其语法如下：

sp_bindefault [@defname =] 'default',
[@objname =] 'object_name'
[, 'futureonly']

其中'futureonly'选项仅在绑定缺省值到用户自定义数据类型上时才可以使用。当指定此选项时，仅以后使用此用户自定义数据类型的列会应用新缺省值，而当前已经使用此数据类型的列则不受影响。

例 9-14：绑定缺省值 today 到用户自定义数据类型 hire_date 上。

exec sp_bindefault today, hire_date

运行结果如下：

---------- ----------

Default bound to data type.

The new default has been bound to columns(s) of the specified user data type.

3．用存储过程 Sp_unbindefault 解除缺省值的绑定

存储过程 Sp_unbindefault 可以解除缺省值与表的列或用户自定义数据类型的绑定，其语法如下：

Sp_unbindefault [@objname =] 'object_name'
[,'futureonly']

其中'futureonly'选项同绑定时一样，仅用于用户自定义数据类型，它指定现有的用此用户自定义数据类型定义的列仍然保持与此缺省值的绑定。如果不指定此项，所有由此用

户自定义数据类型定义的列也将随之解除与此缺省值的绑定。

例 9-15：解除缺省值 num_default 与表 products 的 quantity 列的绑定。

```
exec sp_unbindefault 'products.[quantity]'
```

运行结果如下：

```
---------------------------- ------------------------------
(1 row(s) affected)
Default unbound from table column.
```

> ! 如果列同时绑定了一个规则和一个缺省值，那么缺省值应该符合规则的规定。

> ! 不能绑定缺省值到一个用 CREATE TABLE 或 ALTER TABLE 命令创建或修改表时用 DEFAULT 选项指定了的缺省值的列上。

9.3.4 删除缺省值

可以在 Enterprise Manager 中选择缺省值，单击右键，从快捷菜单中选择“Delete”选项删除缺省值，也可以使用 DROP DEFAULT 命令删除当前数据库中的一个或多个缺省值。其语法如下：

DROP DEFAULT {default_name} [,...n]

> ! 在删除一个缺省值前必须先将与其绑定的对象解除绑定。

例 9-16：删除生日缺省值 birthday_defa。

```
drop default birthday_defa
```

9.4 本章小结

数据完整性工具的应用是 SQL Server 的一大特点，它通过在数据库端使用特定的规定来管理流入与输出系统的信息，而不是由应用程序本身来控制信息的类型，这使得数据独立与应用程序成为开放的数据库系统。