实验 3　数据库完整性控制

知识要点

一、概述

数据库完整性（Database Integrity）是指数据库中数据的正确性和相容性，由各种各样的完整性约束来保证。数据库完整性控制可以通过DBMS或应用程序来实现，基于DBMS的完整性约束作为模式的一部分存入数据库中，其目的在于防止数据库中存在不符合语义的数据，也就是防止数据库中存在不正确的数据，保证数据的语义正确性。

1. 数据库完整性的作用

（1）数据库完整性约束能够防止合法用户使用数据库时向数据库中添加不合语义的数据。

（2）利用基于 DBMS 的完整性控制机制来实现业务规则，易于定义，容易理解，而且可以降低应用程序的复杂性，提高应用程序的运行效率。同时，基于 DBMS 的完整性控制机制是集中管理的，因此比应用程序更容易实现数据库的完整性。

（3）合理的数据库完整性设计，能够同时兼顾数据库的完整性和系统的效能。比如装载大量数据时，只要在装载之前暂时使基于 DBMS 的数据库完整性约束失效，此后再使其生效，就能保证既不影响数据装载的效率又能保证数据库的完整性。

（4）在应用软件的功能测试中，完善的数据库完整性有助于尽早发现应用软件的错误。

2. 数据库完整性分类

（1）实体完整性（Entity Integrity）。

（2）域完整性（Domain Integrity）。

（3）参照完整性（Referential Integrity）。

（4）用户自定义完整性（User-defined Integrity）。

二、实体完整性

1. 实体完整性（Entity Integrity）定义

实体完整性需保证一个表中的每一行必须是唯一的（元组的唯一性）。要保证实体完整性，需指定一个表中的一个属性或一组属性作为它的主键（Primary Key）。一个表中只有一个主键，且一个表中每行的主键必须确实含有一个值。SQL语法中，CREATE TABLE中用PRIMARY KEY定义实体完整性。

（1）单属性构成的主键有两种说明方法。

- 定义为列级约束条件。
- 定义为表级约束条件。

（2）对多个属性构成的主键只有一种说明方法。

- 定义为表级约束条件。

实体完整性规则：每个关系中主键的任何属性不能取空值（注：空值为NULL，不是0，

也不是空格，而是一个“不知道”或“不确定”的数据值）。

2. 实施完整性检查的时机

实施完整性规则检查的时机分为立即检查和延迟检查（Immediate or deferred Checking），只有选择正确的检查时机才能保证语义的正确性，即保证数据的完整性。

例如，“转账”事务的完整性控制条件：转出账户 A 和转入账户 B 的余额之和保持不变。

转账动作：A 减金额，B 加金额。

如果在更新 A 动作发生后立即启动检查就没有意义，应该延迟到 B 更新结束后才检查完整性条件。

实体完整性规则检查的时机是立即检查的，而参照完整性和触发器一般都是延迟检查。

三、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（检查）、NOT NULL（非空）、UNIQUE（唯一）约束都属于域完整性的范畴。

其中CHECK短语不仅能定义属性列上的约束条件，还能定义元组级的约束条件。

四、参照完整性

1. 参照完整性（Referential Integrity）

参照完整性是指两个表的主键和外键数据应对应一致，既可确保表间数据的一致性，又能防止数据丢失或无意义的数据在数据库中存在。

从属的一列或一组列称为外键（Foreign Key）。被引用的列或一组列称之为父键，父键必须是一个主键或唯一键。在 SQL Server 中，在 CREATE TABLE 中用 FOREIGN KEY 短语定义哪些列为外键，用 REFERENCES 短语指明这些外键参照哪些表的主键。

参照完整性规则：关系 R 的外键取值必须是关系 S 中某个元组的主键值，或者可以是一个“空键”。

定义外键时定义参照完整性，约束参照表 A 和被参照表 B。对于违反参照完整性的情况有时候并不是简单拒绝执行，而是接受该操作，同时执行必要的附加操作。DBMS 提供机制来定义是否必须制定外键的具体值而非空值。主键和外键列可以有不同的名字，空值的要求也可以不一致，默认值也可以不同，但数据类型必须相同。

2. SQL Server 中完整性的体现

在 SQL Server 中参照完整性作用表现在以下几个方面：

（1）禁止在参照表中插入包含被参照表中不存在的关键字的数据行。

（2）禁止会导致参照表中相应值孤立的被参照表中外关键字值改变。

（3）禁止删除在参照表中的有对应记录的被参照表记录。

3. SQL 语句中删除和插入基本关系元组

（1）在被参照关系中删除元组的 3 种控制方式：

1）级联删除（CASCADES）：将参照关系中与被参照关系中要删除元组主键值相同的元组一起删除。

2）受限删除（RESTRICTED）：只有参照关系中没有元组与被参照关系中要删除元组主键值相同时才执行删除操作，否则拒绝删除。

3）置空值删除（SET NULL）：删除被参照关系中的元组，同时将参照关系中相应元组的外键值置为空。

（2）在参照关系中插入元组的问题。

1）受限插入：只有被参照关系中有元组与参照关系中要插入元组外键值相同时，才执行插入操作，否则拒绝。

2）递归插入：插入元组外键值在被参照关系中没有元组相同，则首先向被参照关系插入元组，其主键值等于参照关系插入元组的外键值，然后再向参照关系插入元组。

4. DBMS 对参照完整性的检查

（1）在4种情况下DBMS要进行检查，分别是对参照表进行插入和修改以及对被参照表进行删除和修改。

（2）SQL Server在违反参照完整性的4种情况处理方法，如表3-1所示。

表 3-1　SQL Server 四种情况违反参照完整性的处理方法

相关操作	INSERT	DELETE	UPDATE
被参照表	不需要检查	ON DELETE…（用户显示定义的方式，提供两种：cascade 和 no action）	ON UPDATE…（用户显示定义的方式，提供两种：cascade 和 no action）OR default（系统默认的方式 no action）
参照表	拒绝执行	不需要检查	拒绝执行

5. 参照完整性的特殊问题

（1）表的自参照问题。例如：

```
CREAT TABLE employee (
Emp_ id int NOT NULL PRIMARY KEY,
        emp_name varchar(30) NOT NULL,
        Mgr_id int NOT NULL REFERENCES employee (Emp_ id));
```

问题：无法定义。

处理方法：先用CREAT TABLE创建主键约束，再用ALTER TABLE创建外键约束。

（2）两张表互相参照的问题。

问题：无法定义。

处理方法：同表的自参照问题的解决方法相同，详见实验3.2中实验内容9）。

（3）既是外键又是主键中的属性。

处理方法：既要遵从实体完整性也要遵从参照完整性。

五、用户自定义完整性

1. 用户自定义完整性

不同的关系数据库系统根据其应用环境的不同，往往需要一些特殊的约束条件。用户自定义完整性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）。

2. 规则

（1）规则的建立。规则可以是 WHERE 子句中任何有效的表达式，可以包含诸如算术运算符、关系运算符和谓词（如 IN、LIKE、BETWEEN）之类的元素。但不能引用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。

```
CREATE RULE rule AS condition_expression
```

condition_expression 包含一个变量。每个局部变量的前面都有一个 @ 符号。该表达式引用通过 UPDATE 或 INSERT 语句输入的值。在创建规则时，可以使用任何名称或符号表示值，但第一个字符必须是@符号。

（2）规则绑定以及松绑。规则创建后，仅仅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绑定，而一个规则可以绑定多个对象。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）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'：此选项仅当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，当前已使用此数据类型的列则不受影响。

2）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，仅限于用户自定义数据类型。它指定现有的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，则所有由该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。

3. CONSTRAINT 完整性约束命名子句

在定义表时利用约束命名子句对完整性约束条件命名，能够灵活地增加或者删除一个完整性约束条件。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）、唯一性约束（UNIQUE CONSTRAINT）、检查约束（CHECK CONSTRAINT）和缺省约束（DEFAULT CONSTRAINT）。

（1）定义CONSTRAINT约束。

```
CONSTRAINT <完整性约束条件名>
[PRIMARY KEY 短语
   |FOREIGN KEY 短语
   |CHECK 短语]
```

（2）修改 CONSTRAINT 约束。使用 ALTER TABLE 语句修改表中的完整性限制。

4. 触发器

触发器是 SQL Server 数据库应用中的一个重要工具，是一种特殊类型的存储过程，应用非常广泛。一般存储过程主要通过存储过程名而直接被调用，触发器的执行不是由程序调用，也不是手工启动，而是由事件来触发，比如当对一个表进行操作（INSERT、DELETE、UPDATE）时就会激活它执行。当系统规定的触发条件发生时，给定的过程被调用。触发条件是多种多样的，例如：①进入或退出程序的某层结构（如 Block、Form 等）；②查询、修改等操作发生之前或之后；③某个按键动作；④TRIGGER 过程调用（相当于子程序调用）触发器与数据库中的表紧密相关，比如当对表执行 INSERT、UPDATE 或 DELETE 操作时，触发器就会自动执行。触发器经常用于加强数据的完整性约束等。

（1）触发器的类型及两个特殊表。一个触发器只适用于一个表，每个表最多只能有 3 个触发器，它们分别是 INSERT、UPDATE 和 DELETE 触发器。触发器仅在实施数据完整性和处理业务规则时使用。

每个触发器有两个特殊的表，即插入表（INSERTED TABLE）和删除表（DELETED TABLE）。这两个表是逻辑表，并且都是由系统管理，存储在内存中，而非存储于数据库。因此不允许用户直接对其修改，两个表的结构总是与被该触发器作用的表有相同的表结构。

（2）3 种触发器的工作原理。

- INSERT 触发器：先向 INSERTED 表中插入一个新行的副本，然后检查 INSERTED 表中的新行是否有效，确定是否要阻止该插入操作。如果所插入的行中的值有效，则将该行插入到触发器表。
- UPDATE 触发器：先将原始数据行移到 DELETED 表中，然后将一个新行插入 INSERTED 表中，最后计算 DELETED 表和 INSERTED 表中的值以确定是否进行干预。
- DELETE 触发器：将原始数据行移到 DELETED 表中，计算 DELETED 表中的值决定是否进行干预。

（3）SQL 中创建触发器的语法。创建触发器的语法如下：

```
CREATE TRIGGER  <触发器> ON  <表名>
[WITH ENCRYPTION]
FOR {[DELETE][,][INSERT][,][UPDATE]}
[WITH APPEND]
[NOT FOR REPLICATION]
AS <SQL 语句组>
```

注意：创建触发器的语句必须是 SQL 批处理的第一句。

（4）触发器和存储过程的区别。

1）是否附属于唯一的表。触发器附属于唯一的表，而存储过程不附属任何的表。

2）是否事件驱动。触发器由事件驱动，而存储过程由显式的指令调用。

3）是否有数量的限制。一般不允许在表级建立太多的触发器，对触发器的数目有要求，而存储过程没有这方面的要求。

实验 3.1　实体完整性

一、实验目的

1．掌握使用 T-SQL 定义实体完整性的方法。
2．了解 SQL Server 违反实体完整性处理措施。

二、实验内容

1．在数据库 LibraryLib 中重建用户表 Users，分别在列级和表级定义 UserID 为主键。
2．在没有违反实体完整性的前提下向 Users 表中插入并更新一条记录。
3．在 Users 表中验证违反实体完整性的插入操作。
4．在 Users 表中验证违反实体完整性的更新操作。
5．在数据库 LibraryLib 中重建借书表 Borrow，令表中的(UserID,BookID)属性组为主键。
6．验证当与现有的数据环境不等时，无法建立实体完整性。新建数据库 School 并创建学生表 Student，包含以下属性：Sno (CHAR(5))、Sname (CHAR(8))、Ssex (CHAR(1))、Sage (INT)、Sdept (CHAR(20))，插入数据：('10000','王敏','F',23,'CS')，('10000','王浩','M',25,'EE')，令 Sno 为主键，查看结果并分析原因。

三、实验步骤

以系统管理员或sa用户登录进入Microsoft SQL Server Management Studio，新建查询如下：
1．（1）在列级定义主键：

```
USE LibraryLib
DROP TABLE Users
CREATE TABLE  Users(
UserID varchar(20) PRIMARY KEY ,
UserPassword varchar(20) ,
UserPower int  ,
UserName varchar (20) ,
UserSex bit ,
UserDepart varchar(40) ,
UserTelephone varchar(14) ,
UserEMail varchar(30) ,
UserInSystemDate datetime,
UserBorrowedBooks int )
```

（2）在表级定义主键：

```
USE LibraryLib
DROP TABLE Users
CREATE TABLE  Users(
UserID varchar(20) ,
UserPassword varchar(20) ,
UserPower int  ,
```

```
UserName varchar (20) ,
UserSex bit ,
UserDepart varchar(40) ,
UserTelephone varchar(14) ,
UserEMail varchar(30) ,
UserInSystemDate datetime,
UserBorrowedBooks int ,
PRIMARY KEY (UserID))
```

在对象资源管理器中查看主键设置结果，如图 3-1 所示。

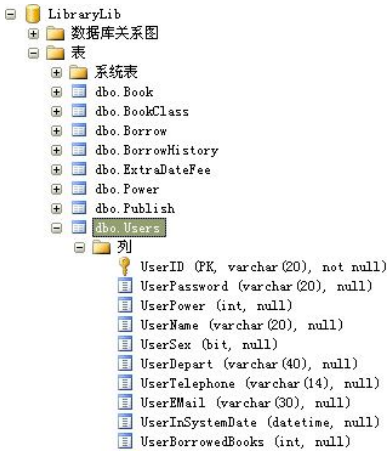

图 3-1　设置主键后 Users 表结构

2.
```
USE LibraryLib
INSERT into Users(
UserID, UserPassword, UserPower, UserName, UserDepart, UserTelephone,
UserEMail)
VALUES('2010072','stt2010',3,'史婷婷','计算机','85213421','stt@126.com');
SELECT * FROM Users;
UPDATE Users SET UserID=' ' WHERE UserName='史婷婷';
SELECT * FROM Users;
UPDATE Users SET UserID=2010072 WHERE UserName='史婷婷';
SELECT * FROM Users;
```

结果如图 3-2 所示。

图 3-2　插入并更新一条记录

思考：成功建立表，令 UserID 为主键。插入与更新操作都没有违反实体完整性。为什么

UserID 置为''，没有违反 NOT NULL 的约束？

3．
```
USE LibraryLib
INSERT into Users(UserID ,UserPassword,UserPower,
UserName,UserDepart,UserTelephone,UserEMail)
VALUES('2010072','hcb2010',4,'贺超波', '计算机','80985221','hcb@126.com');
```
结果如图 3-3 所示。

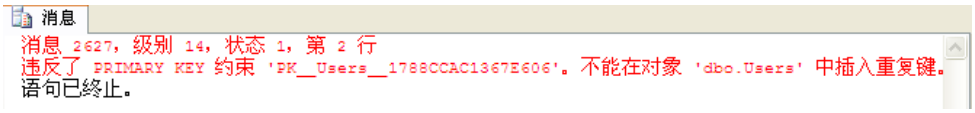

图 3-3　违反实体完整性的插入操作

分析：违反了主键的唯一性属性，将破坏实体完整性，系统中止操作。

4．
```
USE LibraryLib
UPDATE Users SET UserID=NULL WHERE UserName='史婷婷';
```
结果如图 3-4 所示。

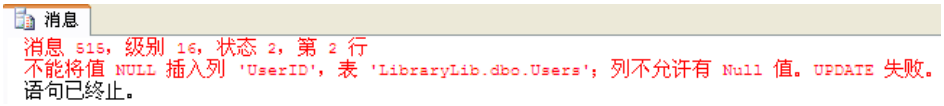

图 3-4　示范违反实体完整性的更新操作

分析：违反了主键的 Not Null 属性，将破坏实体完整性，系统中止操作。

5．
```
USE LibraryLib
DROP TABLE Borrow

CREATE TABLE Borrow(
  UserID varchar(20) NOT NULL,
  BookID int ,
  BorrowBeginDate datetime,
  BorrowEndDate datetime ,
  ManagerID varchar(20),
  PRIMARY KEY(UserID,BookID))
```
总结：关系模型的实体完整性在 CREATE TABLE 中用 PRIMARY KEY 定义。定义主键的方法分为定义为列级约束条件和表级约束条件两种。其中，单属性构成的键有上述两种说明方法，而对多个属性构成的键只有一种说明方法，即定义为表级约束条件。

6．（1）
```
CREATE DATABASE School;
  USE School
  CREATE TABLE Student(
  Sno CHAR(5),
  Sname CHAR(8),
  Ssex CHAR(1),
  Sage INT,
  Sdept CHAR(20));
  INSERT INTO Student values ( '10000','王敏','F',23,'CS');
```

```
INSERT INTO Student values ('10000','王浩','M',25,'EE');
SELECT * FROM Student;
```

结果如图 3-5 所示。

图 3-5　新建数据库 School、创建学生表 Student 并插入记录

（2）
```
USE School
ALTER TABLE Student add
Constraint PK_Student Primary key(Sno)
```

结果如图 3-6 所示。

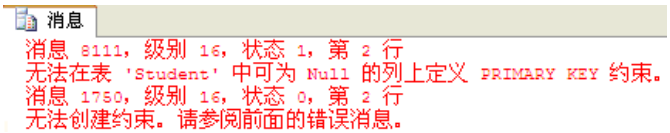

图 3-6　设置 Sno 为主键

分析：当前的数据环境不满足 Sno 成为主键，因为数据列 Sno 不满足实体完整性。

实验 3.2　参照完整性

一、实验目的

1．熟练掌握建立外键的方法。
2．掌握利用 FOREIGN KEY…REFERENCES 子句以及各种约束保证参照完整性。
3．理解参照完整性的含义。

二、实验内容

1．在数据库 LibraryLib 中重建图书表 Book，令 BookID 为主键。

2．修改借书表 Borrow，令 UserID 和 BookID 分别为参照 Users 表以及 Book 表的外键，设定为级联删除。在不违反实体完整性的前提下在向 Users 和 Book 表中插入数据，为下面的实验步骤做预先准备。

3．在不违反参照完整性的前提下，向 Borrow 中插入操作。

4．在 Borrow 表中演示违反参照完整性的插入操作。

5．在 Users 表中删除数据，验证级联删除。

6．在 Book 表中删除数据，验证级联删除。

7．为了验证多重级联删除，在数据库 School 中重建 Student 表，令 Sno 为主键（见实验 3.1 实验步骤 6），并插入数据；新建 StudentCard 表，包含以下属性：CardID (char(14))、Sno(char (5))、RemainedMoney (decimal (10,2))，令 CardID 为其主键，令 Sno 为参照 Student 表的外键，级联删除，并插入数据；新建 ICBCCard 表，包含以下属性：BankID (char(20))、CardID (char

(14))、RestoredMoney (decimal (10,2))，令 BankID 为主键，令 CardID 为参照 StudentCard 表的外键，级联删除，并插入数据。

8．通过删除 Student 表中的一条记录，演示 3 个表的多重级联删除。

9．学校学生会的每个部门都有一个部长，每个部长领导多个学生，每个部只有一个学生有监察评测部长的权力。请给出体现这两种关系（即领导和评测）的两张互参照表的定义。

三、实验步骤

以系统管理员或 sa 用户登录进入 Microsoft SQL Server Management Studio，新建查询如下：

1．
```
USE LibraryLib
DROP TABLE Book
CREATE TABLE Book(
BookID int  PRIMARY KEY,
BookName varchar (50),
BookISBN varchar(20)  ,
BookAuthor varchar(20) ,
BookPublishDate datetime,
BookSubject varchar(30),
BookPrice money,
BookPageNum int,
BookSeries varchar(50),
BookDescription text,
BookNum int,
BookCurNum int,
BookPublishID int,
BookClassID int,
BorrowState int);
```

结果如图 3-7 所示。

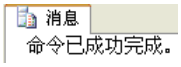

图 3-7　创建图书表 Book

2．
```
USE LibraryLib
ALTER TABLE Borrow  ADD CONSTRAINT FK_Borrow_Users
FOREIGN KEY(UserID) REFERENCES Users(UserID)on delete cascade;
ALTER TABLE Borrow  ADD CONSTRAINT FK_Borrow_Book
FOREIGN KEY(BookID) REFERENCES Book(BookID)on delete cascade;
```

设置外键后，在资源管理器中查看表 Borrow 结构，如图 3-8 所示。

```
INSERT into Users(
  UserID, UserPassword, UserPower, UserName, UserDepart, UserTelephone,
UserEMail)
VALUES('2010071','ym2010',3,'姚明','体育','8233420','ym@163.com');
INSERT into Users(
  UserID, UserPassword, UserPower, UserName, UserDepart, UserTelephone,
UserEMail)
```

图 3-8　对象资源管理器

```
VALUES('2010073','lm2010',3,'李明','体育','8893319','lm@sohu.com');
INSERT INTO Book(BookID,BookName,BookISBN,BookAuthor,BookPublishDate
,BookSubject,BookPrice,BookPageNum,BookSeries)
VALUES(1,'数据库原理及应用','9787508465890','石玉强等',
'2009-08-01 00:00:00.000','数据库',32.0000,311,'21 世纪高等院校规划教材');
INSERT INTO Book(BookID,BookName,BookISBN,BookAuthor,BookPublishDate
,BookSubject,BookPrice,BookPageNum,BookSeries)
VALUES(2,'计算机网络','7115101620','谢希仁',
'2003-05-01 00:00:00.000','计算机网络',39.00,339,'国家级优秀课程');
SELECT * FROM Users;
SELECT * FROM Book;
```

向 Users 和 Book 表中插入记录后结果如图 3-9 所示。

结果　消息

	UserID	UserPassword	UserPower	UserName	UserSex	UserDepart	UserTelephone	UserEMail	UserInSystemDate	UserBorrowedBooks
1	2010071	ym2010	3	姚明	NULL	体育	8233420	ym@163.com	NULL	NULL
2	2010072	stt2010	3	史婷婷	NULL	计算机	85213421	stt@126.com	NULL	NULL
3	2010073	lm2010	3	李明	NULL	体育	8893319	lm@sohu.com	NULL	NULL

结果　消息

	D	BookName	BookISBN	BookAuthor	BookPublishDate	BookSubject	BookPrice	BookPageNum	BookSeries
1		数据库原理及应用	9787508465890	石玉强	2009-08-01 00:00:00.000	数据库	32.00	311	21世纪高等院校规划教材
2		计算机网络	7115101620	谢希仁	2003-05-01 00:00:00.000	计算机网络	39.00	339	国家级优秀课程

图 3-9　向 Users 和 Book 表插入数据

3.
```
USE LibraryLib
INSERT INTO Borrow VALUES('2010071',1,
'2010-03-01 09:40:00.000',
'2010-04-01 09:40:00.000','TTShi')
INSERT INTO Borrow VALUES('2010072',2,
'2010-02-01 15:30:00.000',
'2010-03-01 15:30:00.000','TTShi')
SELECT * FROM Borrow
```

结果如图 3-10 所示。

结果　消息

	UserID	BookID	BorrowBeginDate	BorrowEndDate	ManagerID
1	2010071	1	2010-03-01 09:40:00.000	2010-04-01 09:40:00.000	TTShi
2	2010072	2	2010-02-01 15:30:00.000	2010-03-01 15:30:00.000	TTShi

图 3-10　向 Borrow 表中插入数据

4.
```
USE LibraryLib
INSERT INTO Borrow VALUES('2010074',1,
'2010-03-01 09:40:00.000',
'2010-04-01 09:40:00.000','TTShi')
SELECT * FROM Borrow
```
结果如图 3-11 所示。

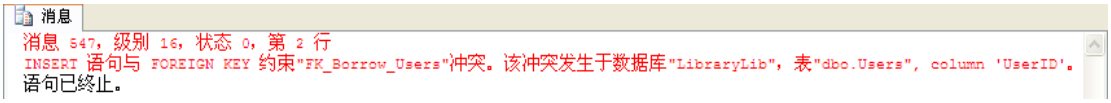

图 3-11　违反了参照完整性的插入操作

分析：违反了参照完整性，表 Users 中没有 UserID 为'2010074'的用户。

5.
```
USE LibraryLib
DELETE FROM Users WHERE UserID='2010071'
SELECT * FROM Borrow
```
结果如图 3-12 所示。

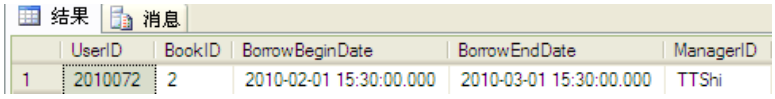

图 3-12　Users 表中级联删除

分析：由于 ON DELETE CASCADE 的级连删除作用，当 Users 表中删除某个 UserID，Borrow 表中对应这个 UserID 为外键的所有记录都要被删除。

6.
```
USE LibraryLib
DELETE FROM Book WHERE BookID='2'
SELECT * FROM Borrow
```
结果如图 3-13 所示。

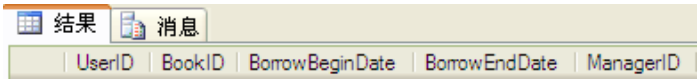

图 3-13　Book 表中级联删除

分析：由于 ON DELETE CASCADE 的级连删除作用，当 Book 表中删除某个 BookID，Borrow 表中对应这个 BookID 为外键的所有记录都要被删除。

7.（1）
```
USE School
DROP TABLE Student
CREATE TABLE Student(
Sno CHAR(5) PRIMARY KEY,
Sname CHAR(8),
Ssex CHAR(1),
Sage INT,
Sdept CHAR(20));
INSERT INTO Student values ( '10000','李敏','F',23,'CS');
INSERT INTO Student values ('10001','王浩','M',25,'EE');
SELECT * FROM Student;
```

结果如图 3-14 所示。

结果　消息

	Sno	Sname	Ssex	Sage	Sdept
1	10000	李敏	F	23	CS
2	10001	王洁	M	25	EE

图 3-14　重建 Student 表并插入数据

分析：利用 DROP TABLE 语句删除 Student 表后再重建。

（2）
```
USE School
CREATE TABLE StudentCard(
CardID char(14) PRIMARY KEY,
Sno char (5) REFERENCES Student(Sno) on delete cascade,
RemainedMoney decimal (10,2)
);
INSERT INTO StudentCard VALUES ( '05212567', '10001',120.00);
INSERT INTO StudentCard VALUES ( '05212302', '10000',130.50);
SELECT * FROM StudentCard;
```

结果如图 3-15 所示。

结果　消息

	CardID	Sno	RemainedMoney
1	05212302	10000	130.50
2	05212567	10001	120.00

图 3-15　新建 StudentCard 表并插入数据

（3）
```
USE School
CREATE TABLE ICBCCard(
BankID char(20) PRIMARY KEY,
CardID char (14) REFERENCES StudentCard(CardID) on delete cascade,
RestoredMoney decimal (10,2),
);
INSERT INTO  ICBCCard VALUES ( '9558844022312','05212302',125000.3);
INSERT INTO  ICBCCard VALUES ( '9558844023645','05212567',150000.4);
SELECT * FROM ICBCCard;
```

结果如图 3-16 所示。

结果　消息

	BankID	CardID	RestoredMoney
1	9558844022312	05212302	125000.30
2	9558844023645	05212567	150000.40

图 3-16　新建 ICBCCard 表并插入数据

8.
```
DELETE FROM Student WHERE Sno='10000';
SELECT * FROM Student;
SELECT * FROM StudentCard;
SELECT * FROM ICBCCard;
```

结果如图 3-17 所示。

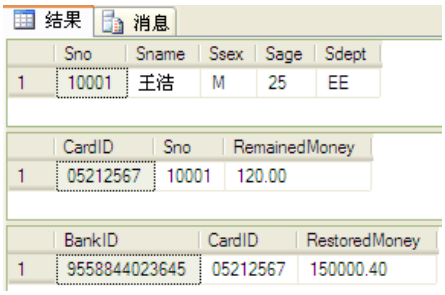

图 3-17　级联删除

分析：数据库 School 中表 StudentCard 和 ICBCCard 的外键均设置为级联删除，故删除表 Student 中某些记录时，系统会自动检查表 StudentCard，若找到相应记录则将它们随之删除；同理表 ICBCCard 中相应记录也随之删除。

9.
```
USE school
CREATE TABLE leader (
Sid char(9), sname varchar(20),myleader char(9)
Constraint PK_leader primary key (sid)
Constraint FK_leader foreign key (myleader) references Monitor(sid)
)
USE school
CREATE TABLE monitor (
Sid char(9),sname varchar(20),mymonitor char(9)
Constraint PK_monitor primary key(sid)
Constraint FK_monitor foreign key (mymonitor) references  Leader(sid)
)
```

结果如图 3-18 所示。

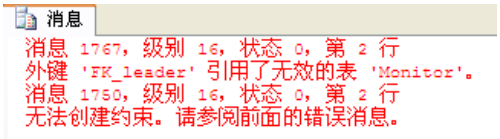

图 3-18　设置外键

解决方法如下：先定义 leader 表，但是不定义外键属性，再定义完整的 monitor 表，用 Alter Table 命令定义 leader 表的外键属性。

（1）
```
USE school
CREATE TABLE leader (
Sid char(9), sname varchar(20),myleader char(9)
Constraint PK_leader primary key (sid)
)
```

（2）
```
USE school
CREATE TABLE monitor (
Sid char(9),sname varchar(20),mymonitor char(9)
```

```
Constraint PK_monitor primary key(sid)
Constraint FK_monitor foreign key (mymonitor) references Leader(sid)
)
ALTER TABLE leader
ADD constraint FK_leader foreign key (myleader) references Monitor(sid)
```

至此，互参照定义完成。

实验 3.3　用户自定义完整性和域完整性

一、实验目的

1．理解用户自定义完整性和域完整性含义。

2．熟练掌握约束、规则实施用户自定义完整性。

3．掌握利用短语 NOT NULL、UNIQUE、CHECK 保证域完整性。

二、实验内容

1．在数据库 School 中新建 Director 表，包含以下属性：Dno (CHAR(5))、Dname (CHAR(8))、Dsex (CHAR(1))、Dage (INT)、Ddept (CHAR(20))，并自定义 3 个约束 U1、U2 和 U3，其中 U1 规定 Dno 字段不允许取空，U2 规定 Dage 属性的值必须≤55，U3 规定 Ddept 属性的值唯一。

2．在 Director 表中插入一条合法记录。

3．分别验证插入违反 U2 和 U3 约束的操作。

4．去除 U2 约束。

5．重新插入 3 中想要插入的数据。

6．创建规则 rule_sex，规定插入或更新的值只能是 M 或 F，并绑定到 Director 的 Dsex 字段。

7．演示违反规则 rule_sex 的插入操作。

8.利用 CONSTRAINT 完整性约束命名子句重建 Student 表(包含属性见实验 3.1 实验步骤 6)，要求 Sno 在 10000～19999 之间，Sname 不能取空值，Ssex 只能是男或女，Sage 小于 30。

9．修改 Student 表中的 CONSTRAINT 完整性约束，去掉对 Ssex 的限制，并将 Sage 的限制由小于 30 改为小于 45，并插入一条合法记录验证是否修改成功。

三、实验步骤

以系统管理员或 sa 用户登录进入 Microsoft SQL Server Management Studio，新建查询如下：

1．
```
USE School
CREATE TABLE Director(
Dno CHAR(5) CONSTRAINT U1 NOT NULL,
Dname CHAR(8) ,
Dsex CHAR(1),
Dage INT CONSTRAINT U2 CHECK (Dage<=55),
Ddept CHAR(20)CONSTRAINT U3 UNIQUE);
```

2.
```
USE School
INSERT INTO Director values ('90000','赵磊','M',45,'艺术系');
SELECT * FROM Director;
```
结果如图 3-19 所示。

结果 消息

	Dno	Dname	Dsex	Dage	Ddept
1	90000	赵磊	M	45	艺术系

图 3-19　Director 表中插入记录

分析：合法插入，没有违反自定义约束。

3.（1）
```
USE School
INSERT INTO Director values ('90001','韩丹','F',60,'中文系');
```
结果如图 3-20 所示。

消息

```
消息 547，级别 16，状态 0，第 2 行
INSERT 语句与 CHECK 约束"U2"冲突。该冲突发生于数据库"School"，表"dbo.Director"，column 'Dage'。
语句已终止。
```

图 3-20　违反了自定义约束 U2 的插入操作

分析：违反了自定义约束 U2，Dage 属性的值必须≤55，插入失败。

（2）
```
USE School
INSERT INTO Director values ('90001','韩丹','F',60,'艺术系');
```
结果如图 3-21 所示。

消息

```
消息 2627，级别 14，状态 1，第 2 行
违反了 UNIQUE KEY 约束 'U3'。不能在对象 'dbo.Director' 中插入重复键。
语句已终止。
```

图 3-21　违反了自定义约束 U3 的插入操作

分析：违反了自定义约束 U3，Ddept 属性的值必须唯一，插入失败。

4.
```
USE School
ALTER TABLE Director DROP U2
```

5.
```
USE School
INSERT INTO Director values ('90001','韩丹','F',60,'中文系');
SELECT * FROM Director;
```
结果如图 3-22 所示。

结果 消息

	Dno	Dname	Dsex	Dage	Ddept
1	90000	赵磊	M	45	艺术系
2	90001	韩丹	F	60	中文系

图 3-22　去除了 U2 约束后的插入操作

分析：去除了自定义约束 U2，所以插入成功。

6.
```
USE school
CREATE RULE rule_sex as @value in ('F','M')
EXEC sp_bindrule rule_sex ,'Director.Dsex';
```
运行结果如图 3-23 所示。

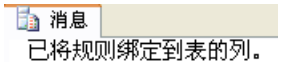

图 3-23　创建规则 rule_sex

在对象资源管理器中查看 School 数据库规则设置，如图 3-24 所示：

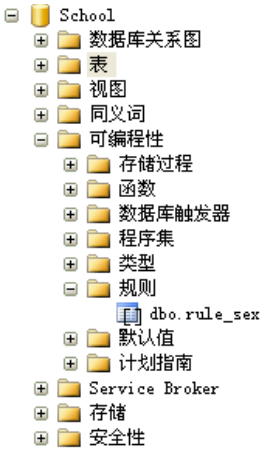

图 3-24　对象资源管理器

分析：设置规则 rule_sex 并绑定到 Director 的 Dsex 字段上。

7.
```
USE School
INSERT INTO Director values ('90002','蒋坤','1',34,'管理系');
```
结果如图 3-25 所示。

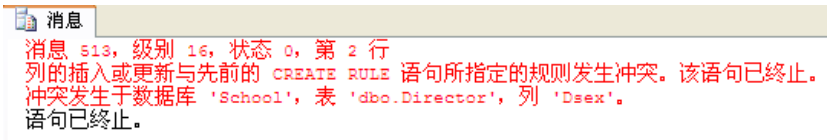

图 3-25　违反规则 rule_sex 的插入操作

分析：插入的数据违反了 sex_rule 规则，操作中止。

8.
```
USE School
Alter TABLE StudentCard DROP FK__StudentCard__Sno__108B795B;
DROP TABLE Student
CREATE TABLE Student(
Sno CHAR(5) CONSTRAINT C1 CHECK(Sno BETWEEN 10000 AND 19999),
Sname CHAR(8)CONSTRAINT C2 NOT NULL,
Ssex CHAR(1)CONSTRAINT C3 CHECK (Ssex IN('男','女')),
Sage INT CONSTRAINT C4 CHECK (Sage<30),
Sdept CHAR(20));
```
注意：由于表 Student 和 StudentCard 间存在外键关联，故删除外键约束后才能成功删除

表 Student。其中外键名称可在对象资源管理器中查看，如图 3-26 所示。

图 3-26　对象资源管理器

9.（1）
```
USE School
  Alter TABLE Student DROP CONSTRAINT  C3 ;
  Alter TABLE Student DROP CONSTRAINT  C4 ;
  Alter TABLE Student ADD CONSTRAINT  C3 CHECK(Sage<40);
```
（2）
```
USE School
INSERT INTO Student VALUES('10002','陈晨','1',39,'播音主持');
SELECT * FROM Student;
```
结果如图 3-27 所示。

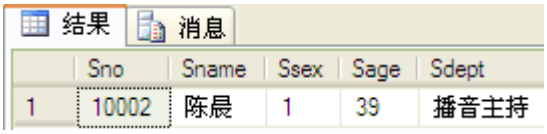

图 3-27　插入一条合法记录

实验 3.4　触发器设计

一、实验目的

1．掌握创建触发器的方法。
2．掌握利用触发器规范插入、更新、删除操作的方法。

二、实验内容

1．在数据库 School 中新建 Teacher 表，包含以下属性：Eno (numeric(4))、Ename (char(10))、Job (char(8))、Sal(numeric(7,2))、Deduct (numeric(7,2))、Deptno(numeric(2))，令 Eno 为主键，建立触发器 T1，当插入或更新表中数据时，保证所操作记录的 Sal 值大于 0，并且教授的工资不得低于 4000 元，如果低于 4000 元，自动改为 4000 元。插入若干条合法记录。
2．验证违反触发器 T1 约束的插入操作。
3．验证违反触发器 T1 约束的更新操作。
4．为 Teacher 表建立触发器 T2，禁止删除 Eno 为 00001 的 Teacher。
5．验证违反触发器 T2 约束的删除操作。

三、实验步骤

以系统管理员或sa用户登录进入Microsoft SQL Server Management Studio，新建查询如下：

1.（1）
```
USE School
CREATE TABLE Teacher
(
Eno numeric(4) primary key,
Ename char(10),
Job char(8),
Sal numeric(7,2),
Deduct numeric(7,2),
Deptno numeric(2));

CREATE TRIGGER Insert_Or_Update_Sal ON Teacher for INSERT,UPDATE
AS
declare @sal numeric(7)
declare @eno numeric(4)
declare @job char(8)
select @sal=Sal from inserted
select @eno=Eno from inserted
select @job=Job from inserted
IF (@sal <1)
BEGIN
Print 'Sage must be a integer mare than zero!'
Rollback transaction
End
ELSE
IF (@job='教授')and (@sal<4000)
BEGIN
select @sal=4000
delete from Teacher where Eno=@eno
insert into Teacher(Eno ,Ename ,Job ,Sal ,Deduct ,Deptno ) select
Eno ,Ename ,Job ,@sal ,Deduct ,Deptno from inserted
END
```

（2）
```
USE School
INSERT INTO Teacher VALUES (1,'林夕','助教',2500,65,1)
INSERT INTO Teacher VALUES (2,'张东华','教授',7000,103,4)
INSERT INTO Teacher VALUES (4,'李列','教授',3000,80,4)
SELECT * FROM Teacher
```

结果如图 3-28 所示。

结果　消息

	Eno	Ename	Job	Sal	Deduct	Deptno
1	1	林夕	助教	2500.00	65.00	1
2	2	张东华	教授	7000.00	103.00	4
3	4	李列	教授	4000.00	80.00	4

图 3-28　插入记录

分析：插入记录(4,'李列','教授',3000,80,4)时，触发 T1 器将其自动改为(4,'李列','教授',4000,80,4)。

2.
```
USE School
INSERT INTO Teacher VALUES (3,'李丽','讲师',-1000,69,1)
```
结果如图 3-29 所示。

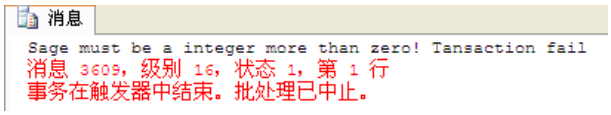

图 3-29　违反 T1 约束的插入操作

分析：插入记录时违反 T1 触发器的约束，操作失败，Rollback!

3.
```
USE School
UPDATE Teacher set Sal=-7 where Eno='1'
```
如图 3-30 所示，在查询分析器中键入 SQL 语句及结果。

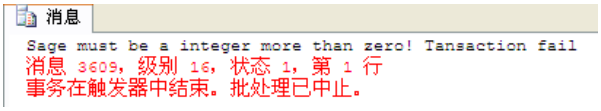

图 3-30　违反 T1 约束的更新操作

分析：更新记录时违反 T1 触发器的约束，操作失败，Rollback!

4.
```
USE school
GO
CREATE TRIGGER T2 on Teacher
FOR DELETE
as
if (select Eno from deleted)='1'
begin
print 'He is the Director!Delete Fail!'
Rollback transaction
End
```

5.
```
USE school
DELETE FROM Teacher where Ename='林夕';
```
结果如图 3-31 所示。

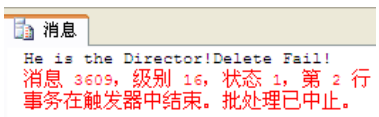

图 3-31　违反 T2 约束的删除操作

分析：删除数据时违反触发器 T2 的约束，操作失败，Rollback!

思考与练习

1. 在 School 数据库中建立一张新表 Class，包括 Class_id(varchar(4))、name(varchar(10))、

Deparment(varchar(20))三个属性，并令 Class_id 为主键。

2．在数据库 LibraryLib 中，用 Alter table 语句将 Borrow 表中的 on delete cascade 改为 on delete restrict，重新插入 Borrow 的数据；重复实验 3.2 实验内容 4、5、6，观察结果，分析原因。

3. 数据库 LibraryLib 中，使用 Alter table 语句将 Borrow 表中的 on delete cascade 改为 on delete set NULL，重新插入 SC 的数据；重复实验 3.2 实验内容 4、5、6，观察结果，分析原因。

4．学校学生会的每个部门都有一个部长，每个部长领导多个学生，每个部只有一个学生有监察评测部长的权力。请给出体现这两种关系（即领导和评测）的两张互参照的表的定义。

5．在数据库 School 中，向 Director 表加入约束 U4，令 Dage >0。

6．承上，去除 U4 约束，加入规则 R1，确保插入的记录的值在 1～60 之间，并绑定到 Dage 属性上。

7．在 School 数据库中，为 Student 表建立触发器 T1，当插入或是更新表中数据时，保证所操作记录的 Sage 值大于 0。

8．建立一个在 Student 表上的触发器 T5，要求当更新一个记录时，表中记录的 Sage 值要比老记录的 Sage 值大，因为一般工资级别只能升不能降。