**安全性和完整性**

**一、实验目的**

熟悉通过SQL对数据库进行数据控制，包括安全性、完整性。

**二、实验环境**

数据库管理系统：SQL Server

交互查询工具：SQL Server Management Studio(SSMS)

**三、实验数据库**

本次实验使用的数据库是自行设计的图书管理系统数据库。该数据库包含以下表：

users：存储系统用户账号、密码和身份。

readers：存储读者详细信息，通过账号关联到users表。

books：存储图书信息。

rb：存储读者的借阅记录，关联readers表和books表。

**四、实验步骤**

**（1）安全性部分**

1.在数据库中由DBA创建若干用户，权限全部选择为CONNECT（SQL Server中的db\_accessadmin角色）

CREATE LOGIN userA WITH PASSWORD = 'Z050818cc@';

CREATE LOGIN userB WITH PASSWORD = 'Z050818cc@';

CREATE LOGIN userC WITH PASSWORD = 'Z050818cc@';

CREATE LOGIN userD WITH PASSWORD = 'Z050818cc@';

USE [library];

GO

CREATE USER userA FOR LOGIN userA;

EXEC sp\_addrolemember 'db\_accessadmin', 'userA';

CREATE USER userB FOR LOGIN userB;

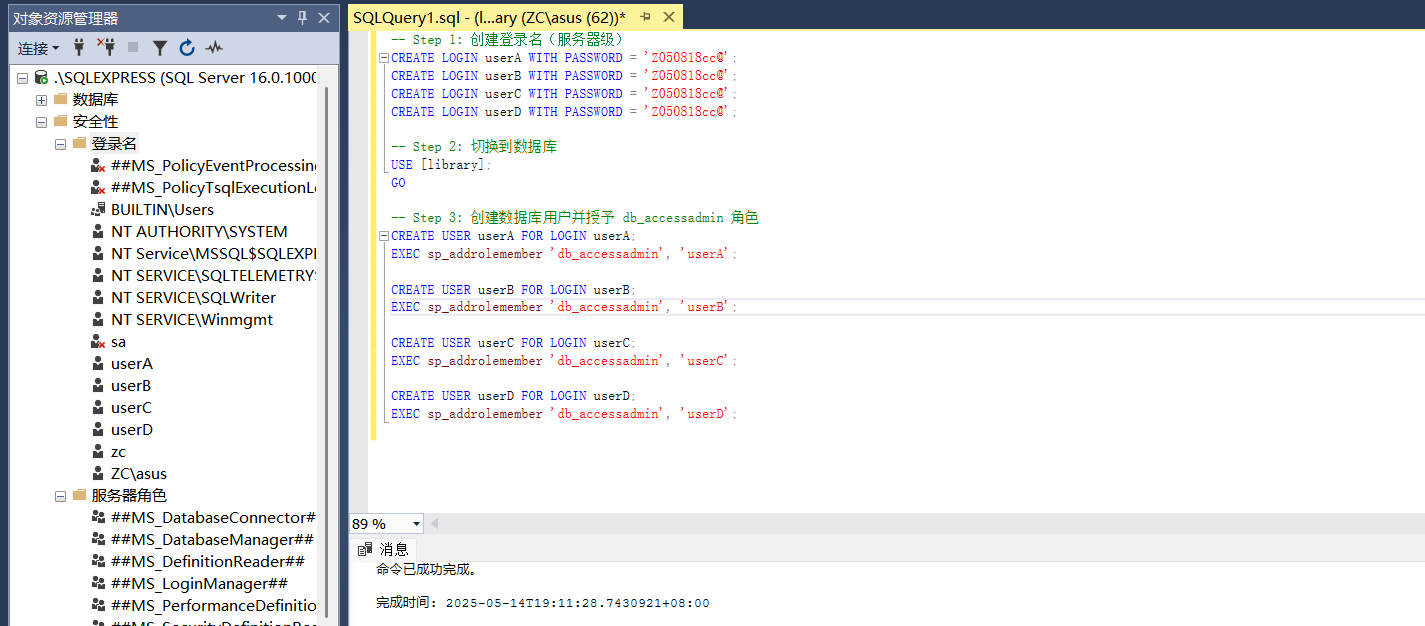
EXEC sp\_addrolemember 'db\_accessadmin', 'userB';

CREATE USER userC FOR LOGIN userC;

EXEC sp\_addrolemember 'db\_accessadmin', 'userC';

CREATE USER userD FOR LOGIN userD;

EXEC sp\_addrolemember 'db\_accessadmin', 'userD';



2.为用户A添加对于books表的select权限。

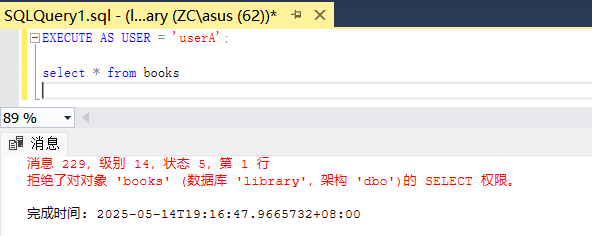
REVERT

GRANT SELECT ON books TO userA;

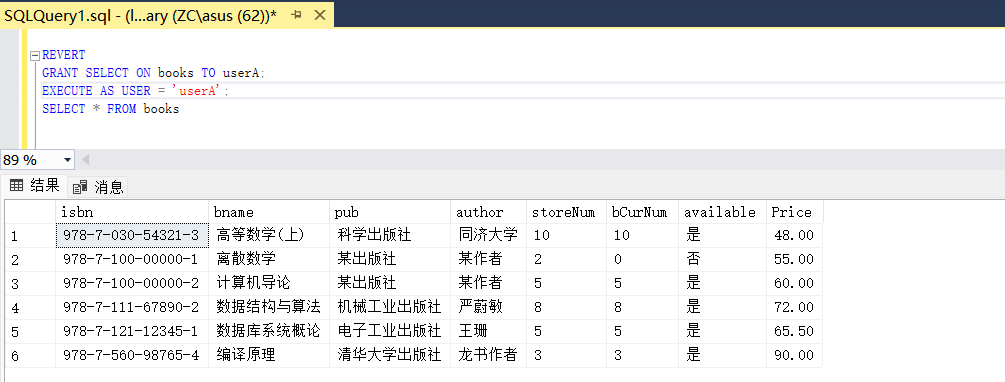
EXECUTE AS USER = 'userA';

SELECT \* FROM books

在添加之前，发现A没有权限进行select操作。



在添加之后，发现A有权限进行select操作。



3.为用户B添加对于books表的所有权限。

REVERT

GRANT SELECT, INSERT, UPDATE, DELETE ON books TO userB;

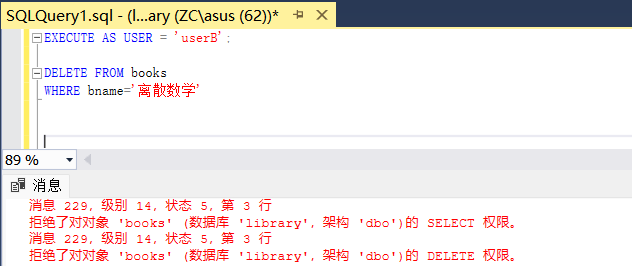
EXECUTE AS USER = 'userB';

DELETE FROM books

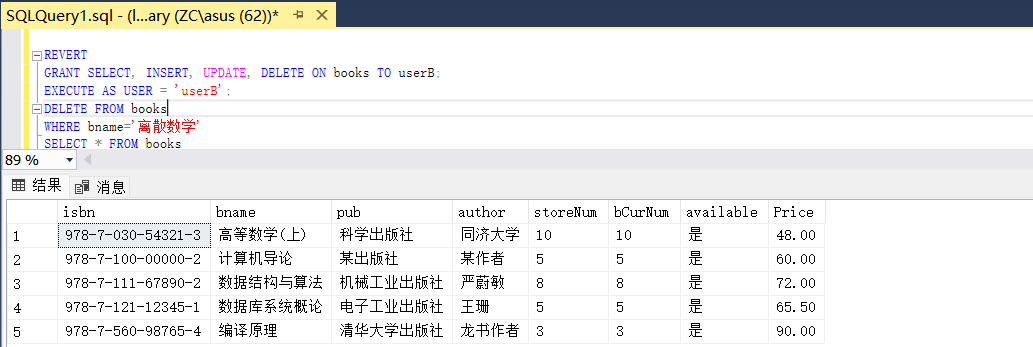
WHERE bname='离散数学'

SELECT \* FROM books

在添加之前，发现B没有权限进行delete操作。



在添加之后，发现B有权限进行delete操作。



4.为用户C添加对于books表的insert权限，并且授权其也可以为其他用户授权insert权限。

REVERT

GRANT INSERT ON books TO userC WITH GRANT OPTION;

EXECUTE AS USER = 'userC';

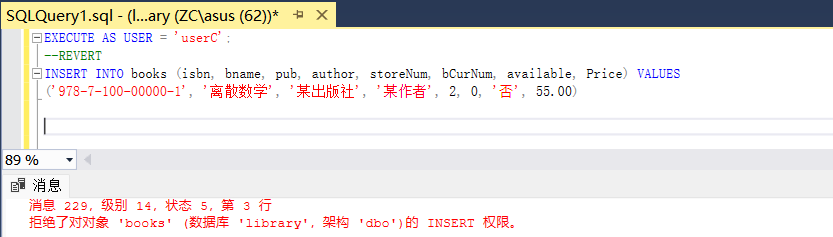
INSERT INTO books (isbn, bname, pub, author, storeNum, bCurNum, available, Price) VALUES

('978-7-100-00000-1', '离散数学', '某出版社', '某作者', 2, 0, '否', 55.00)

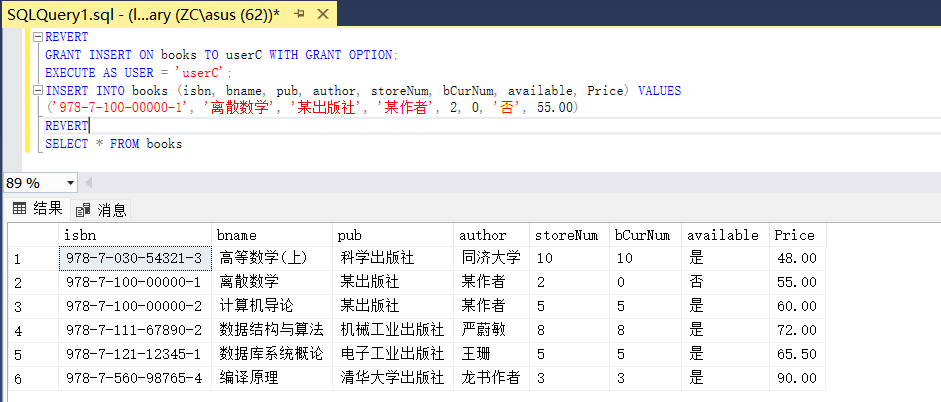
REVERT

SELECT \* FROM books

在添加之前，发现C没有权限进行insert操作。



在添加之后，发现C有权限进行insert操作。



5.为用户D添加对于books表的insert权限，并且授权其也可以为其他用户授权insert权限。

REVERT

EXECUTE AS USER = 'userC';

GRANT INSERT ON books TO userD WITH GRANT OPTION;

REVERT

EXECUTE AS USER = 'userD';

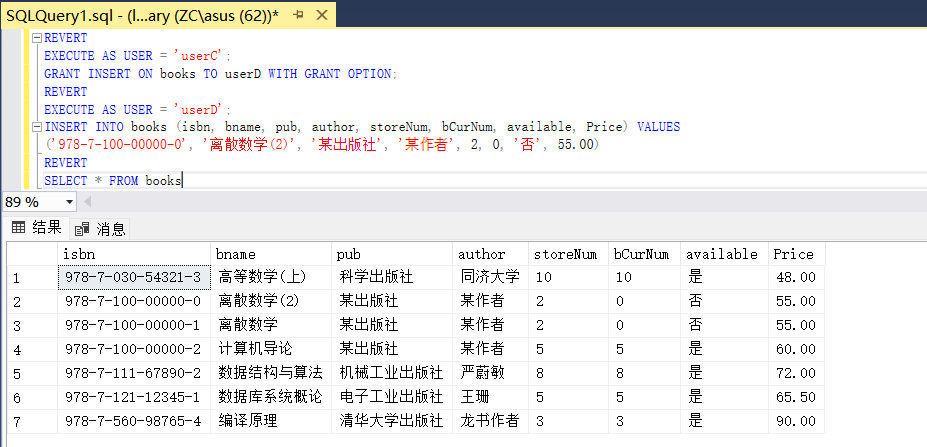
INSERT INTO books (isbn, bname, pub, author, storeNum, bCurNum, available, Price) VALUES

('978-7-100-00000-0', '离散数学(2)', '某出版社', '某作者', 2, 0, '否', 55.00)

REVERT

SELECT \* FROM books

在添加之后，发现D有权限进行insert操作。



6.收回用户A对于books表的select权限。

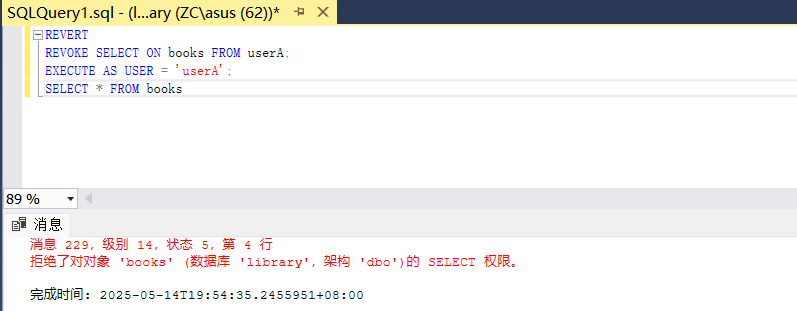
REVERT

REVOKE SELECT ON books FROM userA;

EXECUTE AS USER = 'userA';

SELECT \* FROM books

收回后，发现A没有权限进行select操作。



**（2）完整性部分**

1.实体完整性(仿照[例1]、[例2])x1

例1：在列级定义主码

CREATE TABLE users(

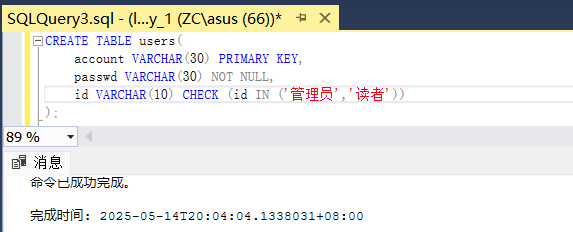
account VARCHAR(30) PRIMARY KEY,

passwd VARCHAR(30) NOT NULL,

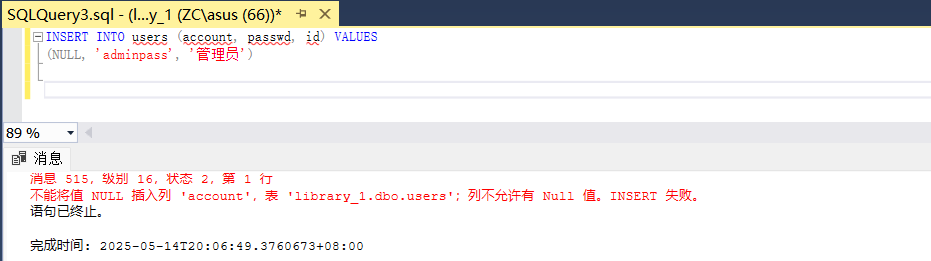
id VARCHAR(10) CHECK (id IN ('管理员','读者'))

);

先创建表，如下图所示创建成功。



若主码account插入值为NULL时，发现报错。



例2：在表级定义主码

CREATE TABLE rb(

borrowid VARCHAR(30),

isbn VARCHAR(30),

startDate DATE,

duration INT,

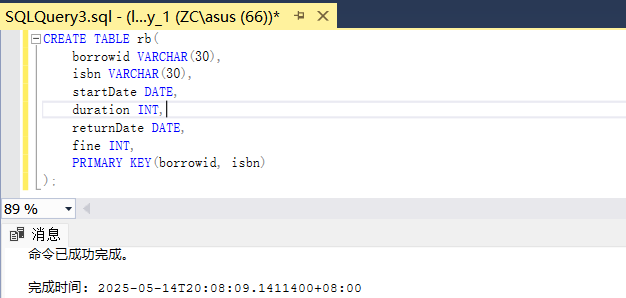
returnDate DATE,

fine INT,

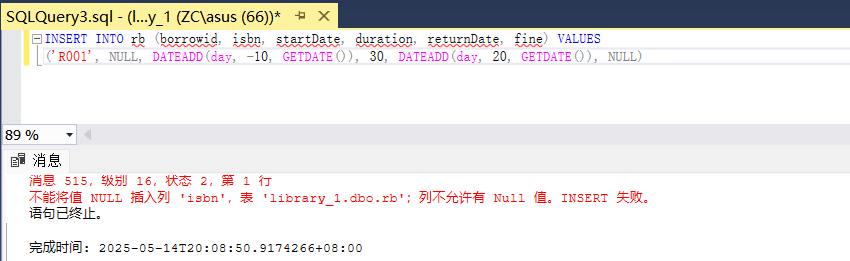
PRIMARY KEY(borrowid, isbn)

);

先创建表，如下图所示创建成功。



若主码（borrowid，isbn）中只要有一个插入值为NULL时，发现报错。



2.参照完整性(仿照[例3])x4

例3：定义参照完整性

①CREATE TABLE readers(

borrowid VARCHAR(30) PRIMARY KEY,

rname VARCHAR(30) NOT NULL,

sex VARCHAR(2) CHECK (sex IN ('男','女')),

job VARCHAR(10),

rCurNum INT,

rBorrowedNum INT,

dept VARCHAR(20),

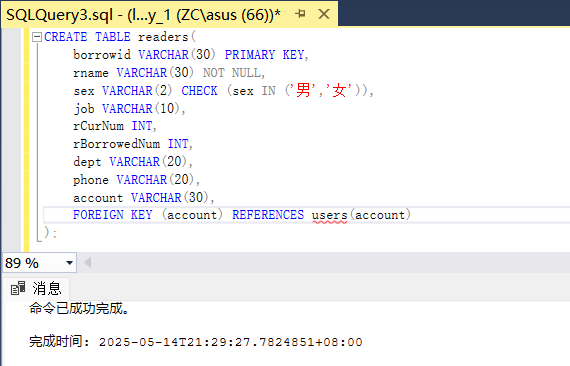
phone VARCHAR(20),

account VARCHAR(30),

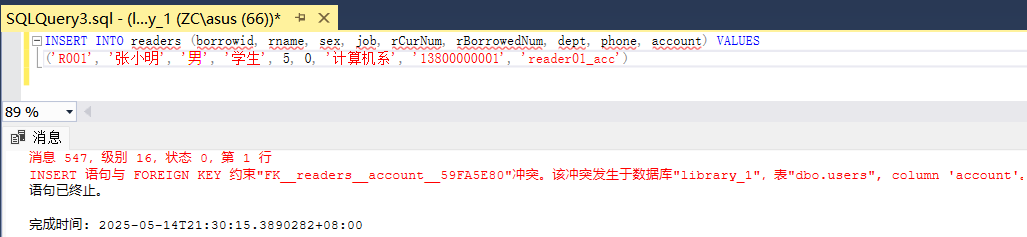
FOREIGN KEY (account) REFERENCES users(account)

);

先创建表，如下图所示创建成功。



因为定义了readers中的account参照了users的account的限制，所以在readers表中尝试插入account不在users中的数据时，发现报错。



②CREATE TABLE rb(

borrowid VARCHAR(30),

isbn VARCHAR(30),

startDate DATE,

duration INT,

returnDate DATE,

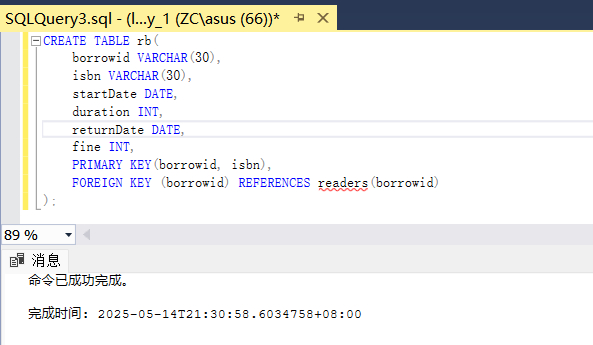
fine INT,

PRIMARY KEY(borrowid, isbn),

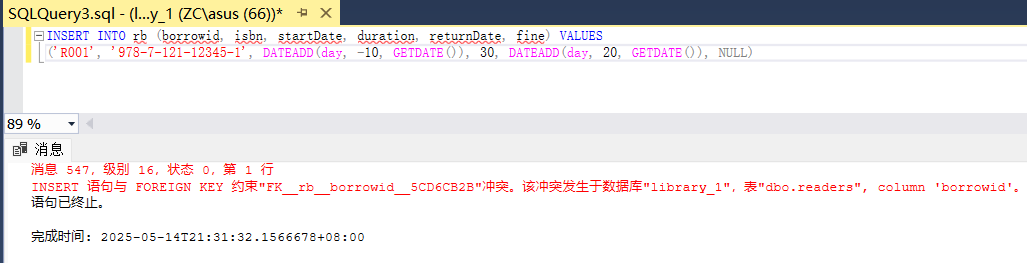
FOREIGN KEY (borrowid) REFERENCES readers(borrowid)

);

先创建表，如下图所示创建成功。



因为定义了rb中的borrowid参照了readers的borrowid的限制，所以在rb表中尝试插入borrowid不在readers中的数据时，发现报错。。



③CREATE TABLE rb(

borrowid VARCHAR(30),

isbn VARCHAR(30),

startDate DATE,

duration INT,

returnDate DATE,

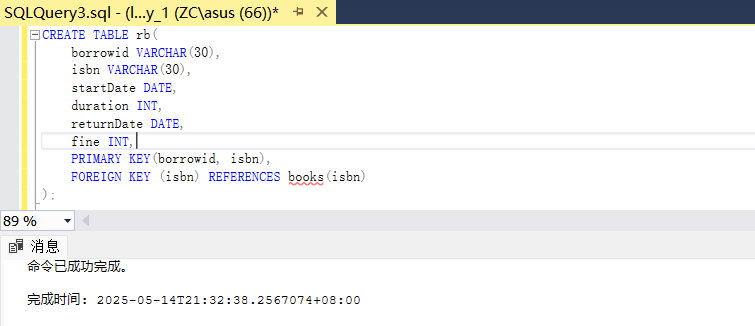
fine INT,

PRIMARY KEY(borrowid, isbn),

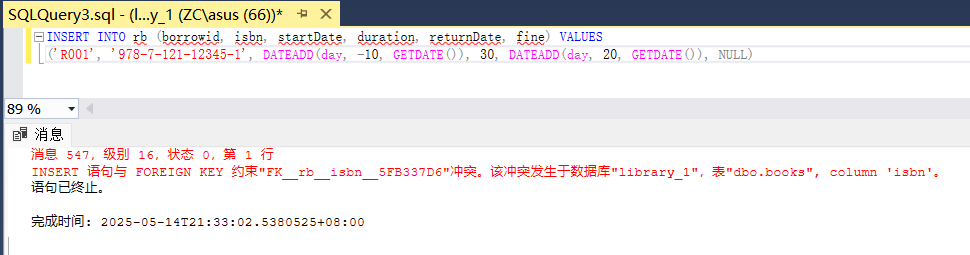
FOREIGN KEY (isbn) REFERENCES books(isbn)

);

先创建表，如下图所示创建成功。



因为定义了rb中的isbn参照了books的isbn的限制，所以在rb表中尝试插入isbn不在books中的数据时，发现报错。



④CREATE TABLE student\_profiles (

profile\_id INT PRIMARY KEY,

student\_id INT,

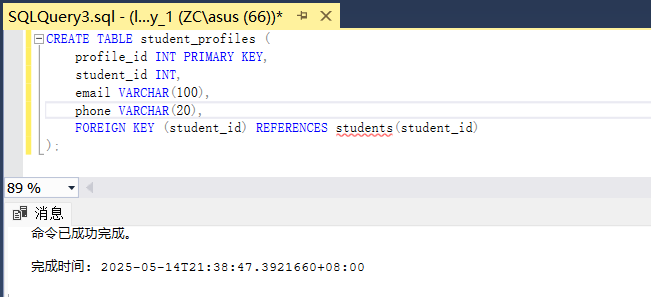
email VARCHAR(100),

phone VARCHAR(20),

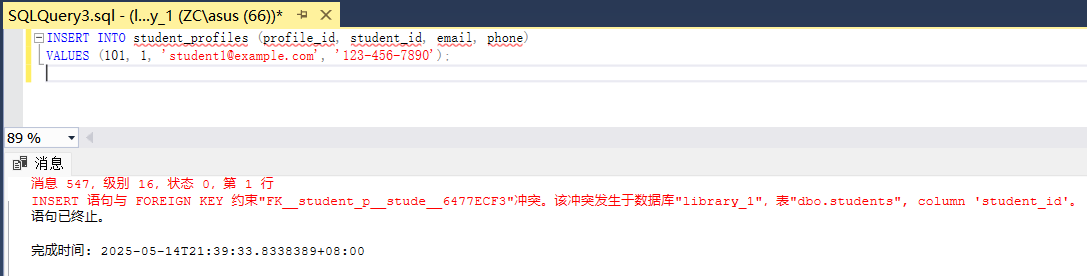
FOREIGN KEY (student\_id) REFERENCES students(student\_id)

);

先创建表，如下图所示创建成功。



因为定义了student\_profiles中的student\_id参照了students的student\_id的限制，所以在student\_profiles表中尝试插入student\_id不在students中的数据时，发现报错。



3.用户定义完整性(仿照[例5]、[例6])x1

例5：不允许取空值

CREATE TABLE users(

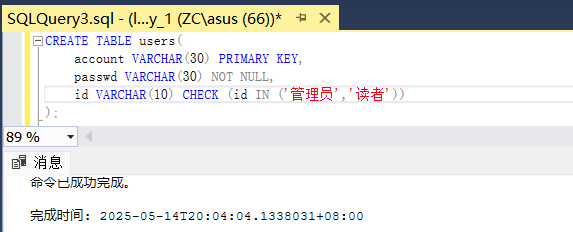
account VARCHAR(30) PRIMARY KEY,

passwd VARCHAR(30) NOT NULL,

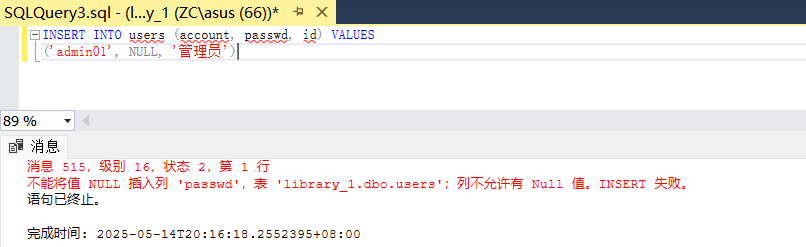
id VARCHAR(10) CHECK (id IN ('管理员','读者'))

);

先创建表，如下图所示创建成功。



因为定义了passwd不能为空值，所以当尝试插入passwd为空值的数据时，发现报错。



例6：列值唯一

CREATE TABLE books(

isbn VARCHAR(30) PRIMARY KEY,

bname VARCHAR(30) UNIQUE,

pub VARCHAR(30) NOT NULL,

author VARCHAR(15) NOT NULL,

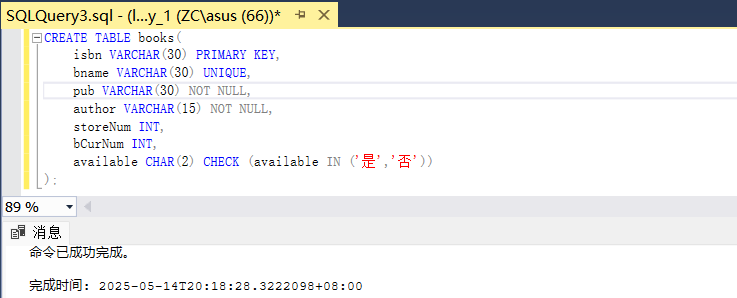
storeNum INT,

bCurNum INT,

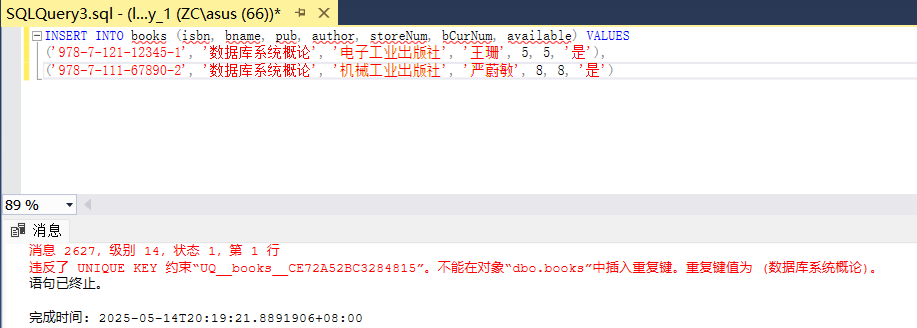
available CHAR(2) CHECK (available IN ('是','否'))

);

先创建表，如下图所示创建成功。



因为定义了bname为唯一值，所以当尝试插入两条bname相同的数据时，发现报错。



4.CHECK短语(仿照[例7]、[例8])x1

例7：用CHECK短语指定列值应该满足的条件

CREATE TABLE users(

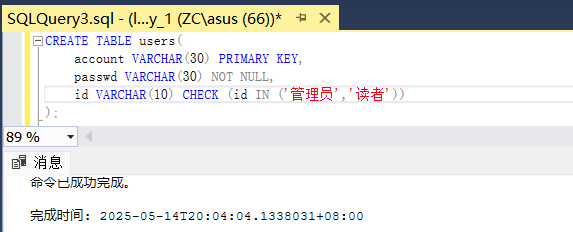
account VARCHAR(30) PRIMARY KEY,

passwd VARCHAR(30) NOT NULL,

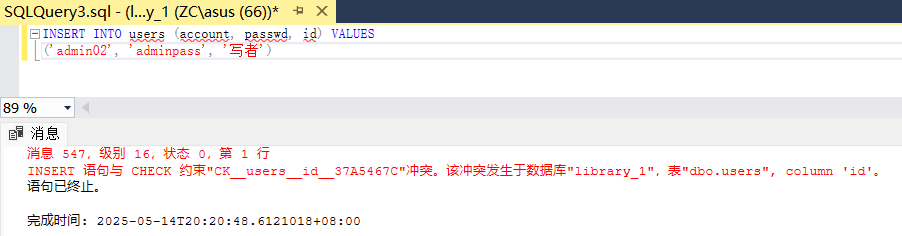
id VARCHAR(10) CHECK (id IN ('管理员','读者'))

);

先创建表，如下图所示创建成功。



因为定义了id必须在（'管理员','读者'）两个值之中，所以当尝试插入id=‘写者’的数据时，发现报错。



例8：用CHECK短语设置元组不同属性间的约束条件

CREATE TABLE books (

isbn VARCHAR(30) PRIMARY KEY,

bname VARCHAR(30) NOT NULL,

pub VARCHAR(30) NOT NULL,

author VARCHAR(15) NOT NULL,

storeNum INT CHECK (storeNum >= 0),

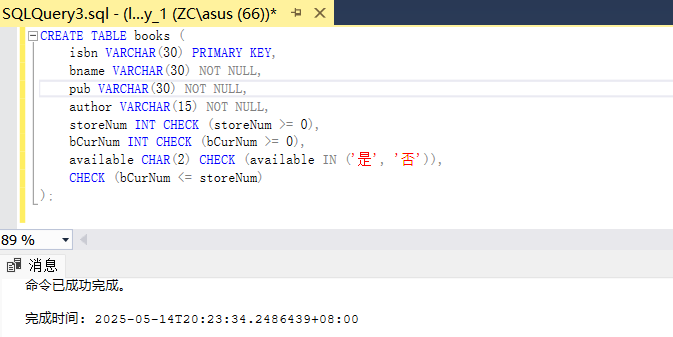
bCurNum INT CHECK (bCurNum >= 0),

available CHAR(2) CHECK (available IN ('是', '否')),

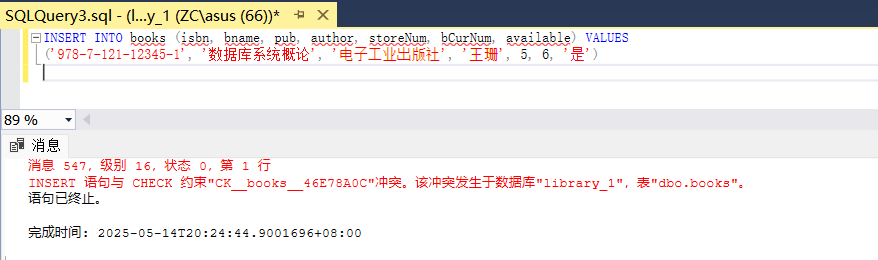
CHECK (bCurNum <= storeNum)

);

先创建表，如下图所示创建成功。



因为定义了bCurNum<=storeNum，所以当尝试插入bCurNum>storeNum的数据时，发现报错。



5.CONSTRAINT子句(仿照[例10]、[例13])x1

例10：使用CONSTRAINT语句命名表中的完整性限制。

CREATE TABLE rb(

borrowid VARCHAR(30),

isbn VARCHAR(30),

CONSTRAINT FK\_rb\_isbn FOREIGN KEY (isbn) REFERENCES books(isbn)

ON DELETE CASCADE

ON UPDATE CASCADE,

startDate DATE,

duration INT,

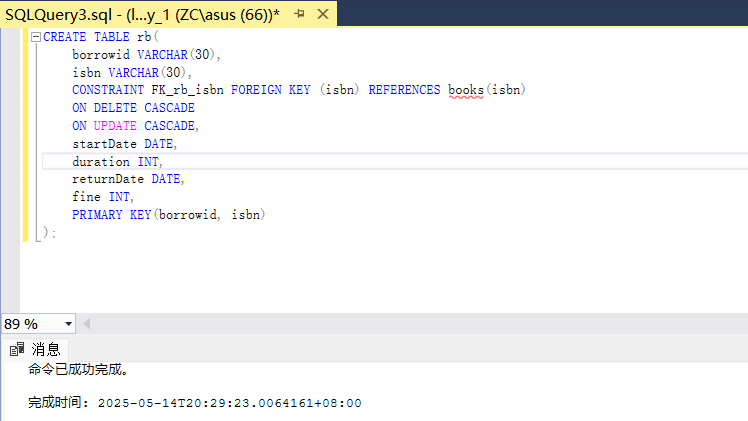
returnDate DATE,

fine INT,

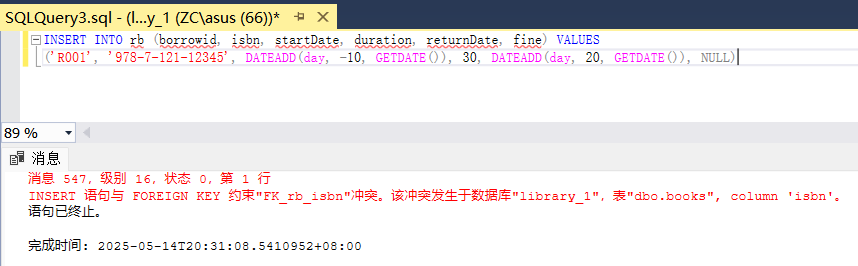
PRIMARY KEY(borrowid, isbn)

);

先创建表，如下图所示创建成功。



因为定义了rb中的isbn参照了books的isbn的限制，所以在rb表中尝试插入isbn不在books中的数据时，发现报错。



例13：使用ALTER TABLE语句修改表中的完整性限制。

CREATE TABLE rb(

borrowid VARCHAR(30),

isbn VARCHAR(30),

CONSTRAINT FK\_rb\_isbn FOREIGN KEY (isbn) REFERENCES books(isbn)

ON DELETE CASCADE

ON UPDATE CASCADE,

startDate DATE,

duration INT,

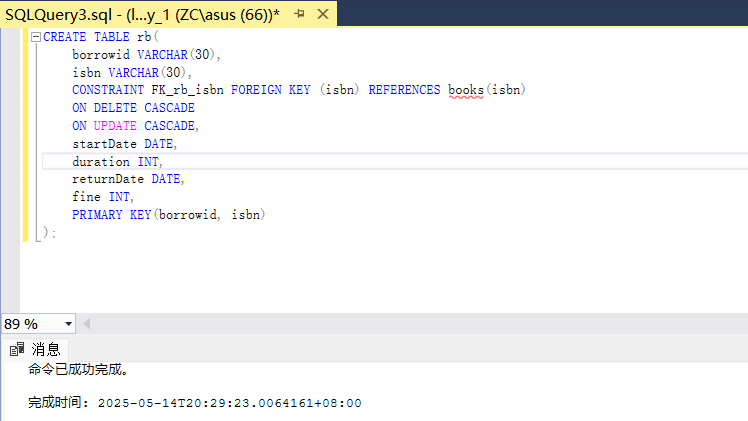
returnDate DATE,

fine INT,

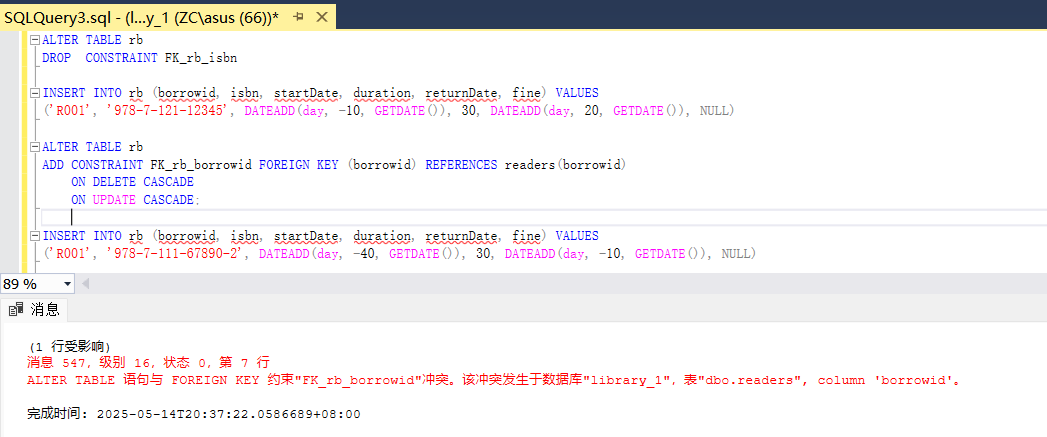
PRIMARY KEY(borrowid, isbn)

);

先创建表，如下图所示创建成功。



对表中的完整性限制进行修改，首先删除了例10中的定义的参照，发现数据可以正常插入了，又添加了一条rb中的borrowid参照了readers的borrowid的限制，所以在rb表中尝试插入borrowid不在readers中的数据时，发现报错。



至此所有实验结束。

**五、出现的问题及解决方法**

实验过程中未出现问题。