

Sql实验-安全性和完整性

姓名	学号	班级	专业
王何佳	2023211603	2023211804	网络空间安全

1. 实验目的

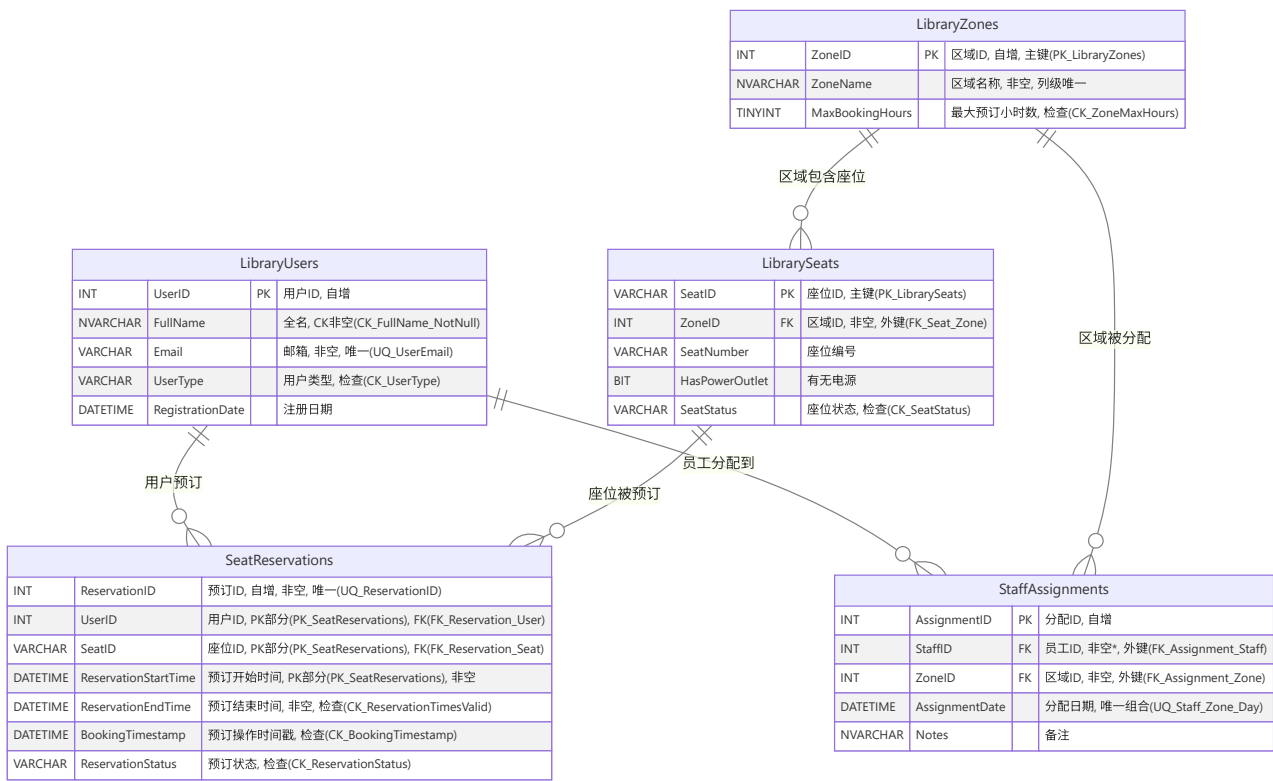
熟悉通过SQL对数据库进行数据控制，包括安全性、完整性和数据库恢复

2. 实验工具

SQL Server

3. 实验数据库

自行设计的图书馆座位预约数据库（含LibraryUsers、LibraryZones、LibrarySeats、SeatReservations、StaffAssignments五个表）



4. 实验内容

4.1 安全性部分

授权与回收

- 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）
- 仿照教材4.2.4 [例1]~[例10]，在DBA与这些用户之间进行授权和回收，并查看效果（4用例）
 - 注意SQL Server中登录名与用户的区别

4.2 完整性部分

使用SQL对数据进行完整性控制，并用实验证实，当操作违反了完整性约束条件时，系统是如何处理的

- 实体完整性(仿照[例1]、[例2]) x1
- 参照完整性(仿照[例3]) x4
- 用户定义完整性(仿照[例5]、[例6]) x1
- CHECK短语(仿照[例7]或 [例8]、[例9]) x1
- CONSTRAINT子句(仿照[例10]、[例13]) x1

5. 实验过程

5.1 安全性部分

5.1.1 创建登录名和用户（CONNECT权限）

创建登录名

```
CREATE LOGIN Login1 WITH PASSWORD = 'P@ss1';
CREATE LOGIN Login2 WITH PASSWORD = 'P@ss2';
CREATE LOGIN Login3 WITH PASSWORD = 'P@ss3';
CREATE LOGIN Login4 WITH PASSWORD = 'P@ss4';
CREATE LOGIN Login5 WITH PASSWORD = 'P@ss5';
CREATE LOGIN Login6 WITH PASSWORD = 'P@ss6';
CREATE LOGIN Login7 WITH PASSWORD = 'P@ss7';
```

安全性

登录名

##MS_PolicyEventProcessingLogin##

##MS_PolicyTsqlExecutionLogin##

Bastandern\62477

Login1

Login2

Login3

Login4

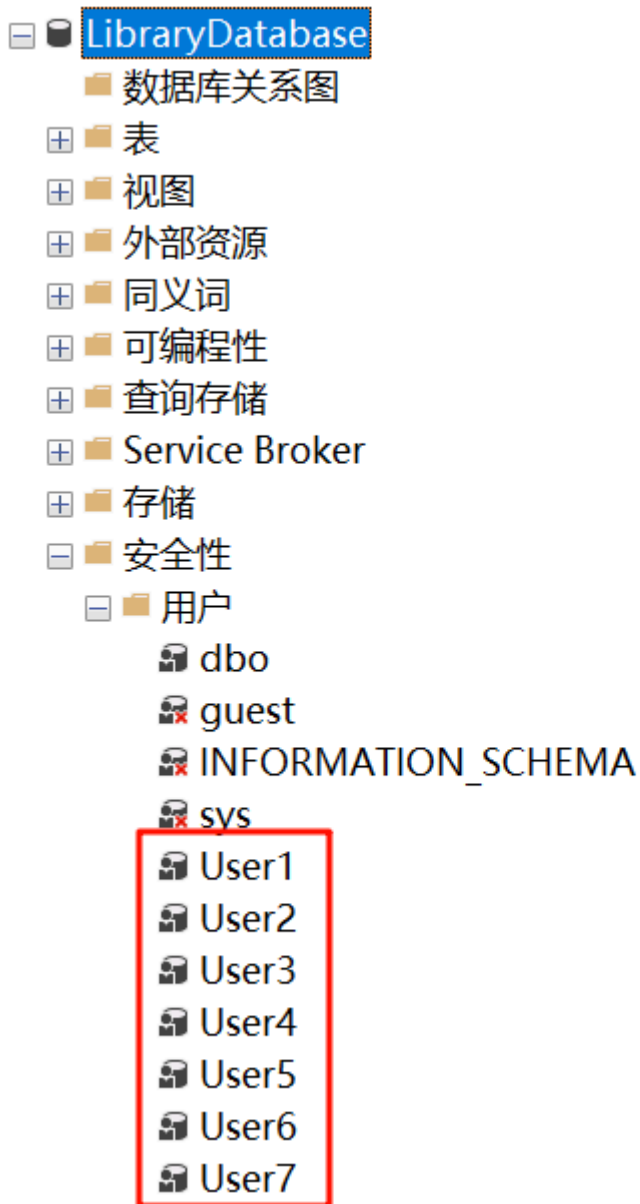
Login5

Login6

Login7

在图书馆座位预约数据库中为这些登录名创建对应的数据库用户

```
Use LibraryDatabase;  
CREATE USER User1 FOR LOGIN Login1;  
CREATE USER User2 FOR LOGIN Login2;  
CREATE USER User3 FOR LOGIN Login3;  
CREATE USER User4 FOR LOGIN Login4;  
CREATE USER User5 FOR LOGIN Login5;  
CREATE USER User6 FOR LOGIN Login6;  
CREATE USER User7 FOR LOGIN Login7;
```



授予这些用户 `CONNECT` 权限

（当用户被创建且其对应的登录名有权访问服务器时，该用户通常已经隐式拥有了 `CONNECT` 权限。此处显式授予以确保）

```
GRANT CONNECT TO User1;  
GRANT CONNECT TO User2;  
GRANT CONNECT TO User3;  
GRANT CONNECT TO User4;  
GRANT CONNECT TO User5;  
GRANT CONNECT TO User6;  
GRANT CONNECT TO User7;
```

5.1.2 授权

(1) 把查询 `LibraryUsers` 表权限授给用户 `User1`

```
GRANT SELECT ON LibraryUsers TO User1;
```

查看 `User1` 属性，成功得到授权

用户名(N): `User1`

安全对象(E):

搜索(S)...

Schema	名称	类型
<code>dbo</code>	<code>LibraryUsers</code>	表

dbo.LibraryUsers 的权限(P):

列权限(C)...

显式

有效

权限	列
SELECT	
SELECT	Email
SELECT	FullName
SELECT	RegistrationDate
SELECT	UserID
SELECT	UserType

使用 `Login1` 登录进行验证

- 查询 `SELECT * FROM LibraryUsers;` 成功
- 查询 `SELECT * FROM LibrarySeats;` 失败

SQLQuery1.sql - B...ase(Login1 (64))*
SELECT * FROM LibraryUsers;

110 %
结果 消息

	UserID	FullName	Email	UserType	RegistrationDate
1	1	张伟	zhangwei@example.com	Staff	2025-04-17 01:25:35.120
2	2	李娜	lina@example.com	Student	2025-04-27 01:25:35.120
3	3	王芳	wangfang@example.com	Faculty	2025-03-18 01:25:35.120
4	4	刘强	liuqian@example.com	Staff	2025-05-07 01:25:35.120
5	5	赵敏	zhaomin@example.com	Student	2025-05-12 01:25:35.120

SQLQuery1.sql - B...ase(Login1 (64))*
SELECT * FROM LibrarySeats;

110 %
消息

消息 229, 级别 14, 状态 5, 第 1 行
拒绝了对对象 'LibrarySeats' (数据库 'LibraryDatabase', 架构 'dbo')的 SELECT 权限。

(2) 把对 **LibraryUsers** 和 **LibrarySeats** 的所有权限授予用户User2和User3
(SQL Server中 **ALL PRIVILEGES** 通常不直接使用, 而是明确列出主要权限, 下面给出这两种方式)

```
GRANT ALL PRIVILEGES ON LibraryUsers TO User2, User3;  
GRANT SELECT, INSERT, UPDATE, DELETE ON LibrarySeats TO User2, User3;
```

查看 **User2** 和 **User3** 属性, 成功得到授权

用户名(N): User2

安全对象(E): 搜索(S)...

Schema	名称	类型
dbo	LibrarySeats	表
dbo	LibraryUsers	表

dbo.LibrarySeats 的权限(P): 列权限(C)...

显式 有效

权限	列
DELETE	
INSERT	
REFERENCES	
REFERENCES	HasPowerOutlet
REFERENCES	SeatID
REFERENCES	SeatNumber
REFERENCES	SeatStatus
REFERENCES	...

用户名(N): User3

安全对象(E): 搜索(S)...

Schema	名称	类型
dbo	LibrarySeats	表
dbo	LibraryUsers	表

dbo.LibrarySeats 的权限(P): 列权限(C)...

显式 有效

权限	列
DELETE	
INSERT	
REFERENCES	
REFERENCES	HasPowerOutlet
REFERENCES	SeatID
REFERENCES	SeatNumber
REFERENCES	SeatStatus
REFERENCES	...

使用 Login3 登录进行验证

SQLQuery1.sql - B...ase(Login3 (53))*

SELECT * FROM LibraryUsers;

110 %

结果 消息

	UserID	FullName	Email	UserType	RegistrationDate
1	1	张伟	zhangwei@example.com	Student	2025-04-17 01:25:35.120
2	2	李娜	lina@example.com	Student	2025-04-27 01:25:35.120
3	3	王芳	wangfang@example.com	Faculty	2025-03-18 01:25:35.120
4	4	刘强	liuqian@example.com	Staff	2025-05-07 01:25:35.120
5	5	赵敏	zhaomin@example.com	Student	2025-05-12 01:25:35.120
6	6	测试用户一	testuser1@example.com	Student	2025-05-17 01:25:35.133
7	7	测试用户二	testuser2@example.com	Student	2025-05-17 01:25:35.133

(3) 把对 SeatReservations 的查询权限授予所有用户

```
GRANT SELECT ON SeatReservations TO PUBLIC;
```

随机登录 Login1、Login3 和 Login7 进行验证，查询成功

```
SELECT * FROM SeatReserveevations;
```

SQLQuery1.sql - B...ase (Login1 (52))*						
SELECT * FROM SeatReservations;						
110 %						
结果 消息						
	ReservationID	UserID	SeatID	ReservationStartTime	ReservationEndTime	BookingTimestamp
1	1000	1	F1Q-A01	2025-05-17 09:00:00.000	2025-05-17 11:00:00.000	2025-05-17 01:25
2	1001	2	GSR201-S1	2025-05-16 14:00:00.000	2025-05-16 15:00:00.000	2025-05-16 01:25
3	1002	1	F3E-C01	2025-05-18 10:00:00.000	2025-05-18 12:00:00.000	2025-05-17 01:25
4	1003	3	F1Q-A02	2025-05-17 09:00:00.000	2025-05-17 10:00:00.000	2025-05-17 01:25

SQLQuery1.sql - B...ase (Login3 (52))*						
SELECT * FROM SeatReservations;						
110 %						
结果 消息						
	ReservationID	UserID	SeatID	ReservationStartTime	ReservationEndTime	BookingTimestamp
1	1000	1	F1Q-A01	2025-05-17 09:00:00.000	2025-05-17 11:00:00.000	2025-05-17 01:25
2	1001	2	GSR201-S1	2025-05-16 14:00:00.000	2025-05-16 15:00:00.000	2025-05-16 01:25
3	1002	1	F3E-C01	2025-05-18 10:00:00.000	2025-05-18 12:00:00.000	2025-05-17 01:25
4	1003	3	F1Q-A02	2025-05-17 09:00:00.000	2025-05-17 10:00:00.000	2025-05-17 01:25

SQLQuery1.sql - B...ase (Login7 (52))*						
SELECT * FROM SeatReservations;						
110 %						
结果 消息						
	ReservationID	UserID	SeatID	ReservationStartTime	ReservationEndTime	BookingTimestamp
1	1000	1	F1Q-A01	2025-05-17 09:00:00.000	2025-05-17 11:00:00.000	2025-05-17 01:25:
2	1001	2	GSR201-S1	2025-05-16 14:00:00.000	2025-05-16 15:00:00.000	2025-05-16 01:25:
3	1002	1	F3E-C01	2025-05-18 10:00:00.000	2025-05-18 12:00:00.000	2025-05-17 01:25:
4	1003	3	F1Q-A02	2025-05-17 09:00:00.000	2025-05-17 10:00:00.000	2025-05-17 01:25:

(4) 把查询 `LibraryUsers` 表和修改该表中 `UserType` 列的权限授予用户 `User4`

```
GRANT UPDATE (UserType), SELECT ON LibraryUsers TO User4;
```


查看 **User4** 属性，成功得到授权

用户名(N): User4

安全对象(E):

搜索(S)...

	Schema	名称	类型
	dbo	LibraryUsers	表

dbo.LibraryUsers 的权限(P):

列权限(C)...

显式有效

权限	列
SELECT	
SELECT	Email
SELECT	FullName
SELECT	RegistrationDate
SELECT	UserID
SELECT	UserType
UPDATE	UserType

使用 Login4 登录进行验证，可以修改

SQLQuery1.sql - B...ase (Login4 (66))*

SELECT * FROM LibraryUsers;

UPDATE LibraryUsers SET UserType = 'Staff' WHERE UserID = 1;

SELECT * FROM LibraryUsers;

110 %

结果 消息

	UserID	FullName	Email	UserType	RegistrationDate
1	1	张伟	zhangwei@example.com	Student	2025-04-17 01:25:35.120
2	2	李娜	lina@example.com	Student	2025-04-27 01:25:35.120
3	3	王芳	wangfang@example.com	Faculty	2025-03-18 01:25:35.120
4	4	刘强	liuqian@example.com	Staff	2025-05-07 01:25:35.120
5	5	赵敏	zhaomin@example.com	Student	2025-05-12 01:25:35.120

	UserID	FullName	Email	UserType	RegistrationDate
1	1	张伟	zhangwei@example.com	Staff	2025-04-17 01:25:35.120
2	2	李娜	lina@example.com	Student	2025-04-27 01:25:35.120
3	3	王芳	wangfang@example.com	Faculty	2025-03-18 01:25:35.120
4	4	刘强	liuqian@example.com	Staff	2025-05-07 01:25:35.120
5	5	赵敏	zhaomin@example.com	Student	2025-05-12 01:25:35.120

(5) 把对 SeatReservations 表的 INSERT 权限授予 User5；User5 将此 INSERT 权限授予 User6，允许 User6 传播；User6 将此 INSERT 权限授予 User7，但不允许 User7 传播。

--DBA操作

GRANT INSERT ON SeatReservations TO User5 WITH GRANT OPTION;

用户名(N): User5

安全对象(E):

搜索(S)...

Schema	名称	类型
dbo	SeatReservations	表

dbo.SeatReservations 的权限(P):

列权限(C)...

显式

有效

权限	列
INSERT	
SELECT	
SELECT	BookingTimestamp
SELECT	ReservationEndTime
SELECT	ReservationID
SELECT	ReservationStartTime
SELECT	ReservationStatus
SELECT	SeatID

以 Login5 身份执行 INSERT ，成功

```
INSERT INTO SeatReservations (
    UserID,
    SeatID,
    ReservationStartTime,
    ReservationEndTime
)
VALUES (
    6,
    'F1Q-A02',
    '2025-05-17 10:00:00',
    '2025-05-17 12:00:00'
);
```

结果

消息

	ReservationID	UserID	SeatID	ReservationStartTime	ReservationEndTime	BookingTimestamp
1	1000	1	F1Q-A01	2025-05-17 09:00:00.000	2025-05-17 11:00:00.000	2025-05-17 01:25:35.150
2	1001	2	GSR201-S1	2025-05-16 14:00:00.000	2025-05-16 15:00:00.000	2025-05-16 01:25:35.150
3	1002	1	F3E-C01	2025-05-18 10:00:00.000	2025-05-18 12:00:00.000	2025-05-17 01:25:35.150
4	1003	3	F1Q-A02	2025-05-17 09:00:00.000	2025-05-17 10:00:00.000	2025-05-17 01:25:35.150
5	1004	6	F1Q-A02	2025-05-17 10:00:00.000	2025-05-17 12:00:00.000	2025-05-17 02:22:24.650

再将权限授予 `User6`，允许传播

```
GRANT INSERT ON SeatReservations TO User6 WITH GRANT OPTION;
```

以 `Login6` 身份执行 `INSERT`，成功

```
INSERT INTO SeatReservations (
    UserID,
    SeatID,
    ReservationStartTime,
    ReservationEndTime
)
VALUES (
    7,
    'GSR201-S1',
    '2025-05-18 14:00:00',
    '2025-05-18 15:30:00'
);
```

结果

消息

	ReservationID	UserID	SeatID	ReservationStartTime	ReservationEndTime	BookingTimestamp
1	1000	1	F1Q-A01	2025-05-17 09:00:00.000	2025-05-17 11:00:00.000	2025-05-17 01:25:35.150
2	1001	2	GSR201-S1	2025-05-16 14:00:00.000	2025-05-16 15:00:00.000	2025-05-16 01:25:35.150
3	1002	1	F3E-C01	2025-05-18 10:00:00.000	2025-05-18 12:00:00.000	2025-05-17 01:25:35.150
4	1003	3	F1Q-A02	2025-05-17 09:00:00.000	2025-05-17 10:00:00.000	2025-05-17 01:25:35.150
5	1004	6	F1Q-A02	2025-05-17 10:00:00.000	2025-05-17 12:00:00.000	2025-05-17 02:22:24.650
6	1005	7	GSR201-S1	2025-05-18 14:00:00.000	2025-05-18 15:30:00.000	2025-05-17 02:27:40.000

再将权限授予 `User7`，不允许传播

```
GRANT INSERT ON SeatReservations TO User7;
```

以 `Login7` 身份执行 `INSERT`，成功

```
INSERT INTO SeatReservations (
    UserID,
    SeatID,
    ReservationStartTime,
    ReservationEndTime,
    ReservationStatus
);
```

```

)
VALUES (
    8,
    'F3E-C01',
    '2025-05-19 09:00:00',
    '2025-05-19 11:30:00',
    'Confirmed'
);

```

	ReservationID	UserID	SeatID	ReservationStartTime	ReservationEndTime	BookingTimestamp
1	1000	1	F1Q-A01	2025-05-17 09:00:00.000	2025-05-17 11:00:00.000	2025-05-17 01:25:35.
2	1001	2	GSR201-S1	2025-05-16 14:00:00.000	2025-05-16 15:00:00.000	2025-05-16 01:25:35.
3	1002	1	F3E-C01	2025-05-18 10:00:00.000	2025-05-18 12:00:00.000	2025-05-17 01:25:35.
4	1003	3	F1Q-A02	2025-05-17 09:00:00.000	2025-05-17 10:00:00.000	2025-05-17 01:25:35.
5	1004	6	F1Q-A02	2025-05-17 10:00:00.000	2025-05-17 12:00:00.000	2025-05-17 02:22:24.
6	1005	7	GSR201-S1	2025-05-18 14:00:00.000	2025-05-18 15:30:00.000	2025-05-17 02:27:48.
7	1006	8	F3E-C01	2025-05-19 09:00:00.000	2025-05-19 11:30:00.000	2025-05-17 02:31:00.

再将权限授予 **User1**，失败

```
GRANT INSERT ON SeatReservations TO User1;
```

110 %

消息

消息 15151, 级别 16, 状态 1, 第 1 行
无法对 对象 'SeatReservations' 执行 查找, 因为它不存在, 或者您没有所需的权限。

5.1.3 回收

(1) 把用户 **User4** 修改 **LibraryUsers** 表 **UserType** 列的权限收回

```
REVOKE UPDATE (UserType) ON LibraryUsers FROM User4;
```

查看 **User4** 属性，已经没有 **UPDATE** 权限

用户名(N):

安全对象(E):

Schema	名称	类型
dbo	LibraryUsers	表

dbo.LibraryUsers 的权限(P):

权限	列
SELECT	
SELECT	Email
SELECT	FullName
SELECT	RegistrationDate
SELECT	UserID
SELECT	UserType

以 **Login4** 身份执行 **UPDATE**，失败；执行 **SELECT** 仍然成功

SQLQuery1.sql - B...ase (Login4 (53))*

```
UPDATE LibraryUsers SET UserType = 'Student' WHERE UserID = 1;
```

110 %

消息

消息 229, 级别 14, 状态 5, 第 1 行
拒绝了对对象 'LibraryUsers' (数据库 'LibraryDatabase', 架构 'dbo')的 UPDATE 权限。

SQLQuery1.sql - B...ase Login4 (53))*

```
SELECT * FROM LibraryUsers;
```

110 %

结果 消息

	UserID	FullName	Email	UserType	RegistrationDate
1	1	张伟	zhangwei@example.com	Staff	2025-04-17 01:25:35.120
2	2	李娜	lina@example.com	Student	2025-04-27 01:25:35.120
3	3	王芳	wangfang@example.com	Faculty	2025-03-18 01:25:35.120
4	4	刘强	liuqian@example.com	Staff	2025-05-07 01:25:35.120
5	5	赵敏	zhaomin@example.com	Student	2025-05-12 01:25:35.120

(2) 收回所有用户对表 `SeatReservations` 的查询权限

```
REVOKE SELECT ON SeatReservations FROM PUBLIC;
```

随机登录 `Login2`、`Login3` 和 `Login7` 进行验证，查询失败

SQLQuery1.sql - B...ase (Login2 (64))*

```
SELECT * FROM SeatReservations;
```

110 %

消息

消息 229, 级别 14, 状态 5, 第 1 行
拒绝了对对象 'SeatReservations' (数据库 'LibraryDatabas

SQLQuery1.sql - B...ase (Login3 (64))*

```
SELECT * FROM SeatReservations;
```

110 %

消息

消息 229, 级别 14, 状态 5, 第 1 行
拒绝了对对象 'SeatReservations' (数据库 'LibraryDatabas

SQLQuery1.sql - B...ase (Login7 (82))*

```
SELECT * FROM SeatReservations;
```

110 %

消息

消息 229, 级别 14, 状态 5, 第 1 行
拒绝了对对象 'SeatReservations' (数据库 'LibraryDatabas

(3) 把用户 **User5** 对 **SeatReservations** 表的 **INSERT** 权限及其通过 **WITH GRANT OPTION** 传播出去的权限一并收回

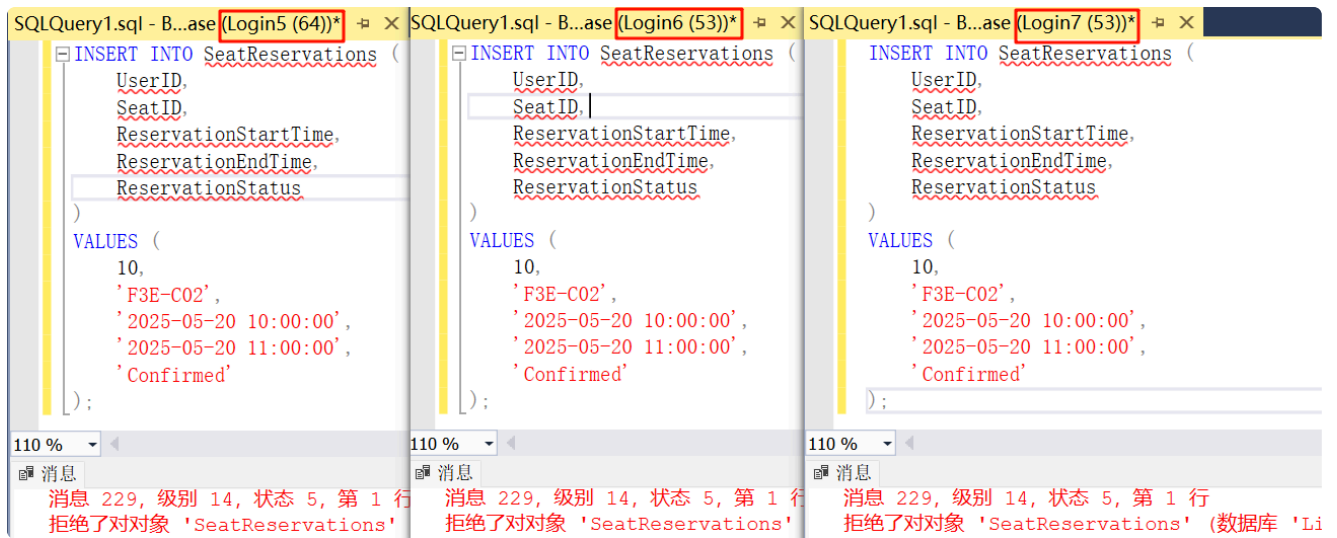
```
REVOKE INSERT ON SeatReservations FROM User5 CASCADE;
```

查看 **User5**、**User6**、**User7** 属性，已经没有任何权限

用户名(N):	User5	用户名(N):	User6	用户名(N):	User7
安全对象(E):	Schema	安全对象(E):	Schema	安全对象(E):	Schema
权限(P):	显式 权限	权限(P):	显式 权限	权限(P):	显式 权限

以 **User5**、**User6**、**User7** 身份执行 **UPDATE**，均失败

```
INSERT INTO SeatReservations (  
    UserID,  
    SeatID,  
    ReservationStartTime,  
    ReservationEndTime,  
    ReservationStatus  
)  
VALUES (  
    10,  
    'F3E-C02',  
    '2025-05-20 10:00:00',  
    '2025-05-20 11:00:00',  
    'Confirmed'  
);
```

5.2 完整性部分

使用的建库代码如下，接下来我将分析其中包含的完整性约束条件。

```
-- LibraryUsers (用户信息表)
CREATE TABLE LibraryUsers (
    UserID INT IDENTITY(1,1) PRIMARY KEY,          -- 用户ID, 主键, 自增
    FullName NVARCHAR(100),                        -- 用户全名
    Email VARCHAR(100) NOT NULL,                   -- 邮箱
    UserType VARCHAR(20) DEFAULT 'Student',        -- 用户类型
    RegistrationDate DATETIME DEFAULT GETDATE(),   -- 注册日期
    CONSTRAINT CK_FullName_NotNull CHECK (FullName IS NOT NULL)
    CONSTRAINT UQ_UserEmail UNIQUE (Email),
    CONSTRAINT CK_UserType
        CHECK (UserType IN ('Student', 'Faculty', 'Staff'))
);

-- LibraryZones (图书馆区域信息表)
CREATE TABLE LibraryZones (
    ZoneID INT IDENTITY(1,1),                      -- 区域ID, 自增
    ZoneName NVARCHAR(50) NOT NULL UNIQUE,         -- 区域名称
    MaxBookingHours TINYINT DEFAULT 2,             -- 允许的最大预订小时数
    CONSTRAINT PK_LibraryZones PRIMARY KEY (ZoneID),
    CONSTRAINT CK_ZoneMaxHours
        CHECK (MaxBookingHours BETWEEN 1 AND 4)
);

-- LibrarySeats (图书馆座位信息表)
CREATE TABLE LibrarySeats (
    SeatID VARCHAR(10),                            -- 座位ID
    ZoneID INT NOT NULL,                            -- 所属区域ID
    SeatNumber VARCHAR(5) NULL,                     -- 座位在区域内的编号
    HasPowerOutlet BIT DEFAULT 0,                  -- 是否有电源插座
    SeatStatus VARCHAR(15) DEFAULT 'Available',    -- 座位状态
    CONSTRAINT PK_LibrarySeats PRIMARY KEY (SeatID),
```

```

CONSTRAINT FK_Seat_Zone FOREIGN KEY (ZoneID)
    REFERENCES LibraryZones(ZoneID)
    ON DELETE NO ACTION
    ON UPDATE CASCADE,
CONSTRAINT CK_SeatStatus
    CHECK (SeatStatus IN ('Available', 'Maintenance'))
);

-- SeatReservations (座位预订信息表)
CREATE TABLE SeatReservations (
    ReservationID INT IDENTITY(1000,1) NOT NULL, -- 预订ID, 自增
    UserID INT NOT NULL, -- 用户ID
    SeatID VARCHAR(10) NOT NULL, -- 座位ID
    ReservationStartTime DATETIME NOT NULL, -- 预订开始时间
    ReservationEndTime DATETIME NOT NULL, -- 预订结束时间
    BookingTimestamp DATETIME DEFAULT GETDATE(), -- 预订操作时间戳
    ReservationStatus VARCHAR(15) DEFAULT 'Confirmed', -- 预订状态
    CONSTRAINT UQ_ReservationID UNIQUE (ReservationID),
    CONSTRAINT PK_SeatReservations
        PRIMARY KEY (UserID, SeatID, ReservationStartTime),
    CONSTRAINT FK_Reservation_User FOREIGN KEY (UserID)
        REFERENCES LibraryUsers(UserID)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT FK_Reservation_Seat FOREIGN KEY (SeatID)
        REFERENCES LibrarySeats(SeatID)
        ON DELETE NO ACTION
        ON UPDATE CASCADE,
    CONSTRAINT CK_ReservationTimesValid
        CHECK (ReservationEndTime > ReservationStartTime),
    CONSTRAINT CK_ReservationStatus
        CHECK (ReservationStatus IN ('Confirmed', 'Cancelled', 'Completed')),
    CONSTRAINT CK_BookingTimestamp
        CHECK (BookingTimestamp <= ReservationStartTime)
);

-- StaffAssignments (员工区域分配表)
CREATE TABLE StaffAssignments (
    AssignmentID INT IDENTITY(1,1) PRIMARY KEY, -- 分配ID
    StaffID INT NULL, -- 员工ID
    ZoneID INT NOT NULL, -- 区域ID
    AssignmentDate DATETIME DEFAULT GETDATE(), -- 分配日期
    Notes NVARCHAR(255) NULL, -- 备注

    CONSTRAINT FK_Assignment_Staff FOREIGN KEY (StaffID)
        REFERENCES LibraryUsers(UserID)
        ON DELETE NO ACTION
        ON UPDATE CASCADE,

    CONSTRAINT FK_Assignment_Zone FOREIGN KEY (ZoneID)

```

```
REFERENCES LibraryZones(ZoneID)
ON DELETE NO ACTION
ON UPDATE CASCADE,

CONSTRAINT UQ_Staff_Zone_Day
    UNIQUE (StaffID, ZoneID, AssignmentDate)
);
```

数据库信息如下：

结果		LibraryUsers				
	UserID	FullName	Email	UserType	RegistrationDate	
1	1	张伟	zhangwei@example.com	Staff	2025-04-17 01:25:35.120	
2	2	李娜	lina@example.com	Student	2025-04-27 01:25:35.120	
3	3	王芳	wangfang@example.com	Faculty	2025-03-18 01:25:35.120	
4	4	刘强	liuqiang@example.com	Staff	2025-05-07 01:25:35.120	
5	5	赵敏	zhaomin@example.com	Student	2025-05-12 01:25:35.120	
6	6	测试用户一	testuser1@example.com	Student	2025-05-17 01:25:35.133	
7	7	测试用户二	testuser2@example.com	Student	2025-05-17 01:25:35.133	
8	8	测试用户三	testuser3@example.com	Faculty	2025-05-17 01:25:35.133	
9	9	测试用户四	testuser4@example.com	Staff	2025-05-17 01:25:35.133	
10	10	测试用户五	testuser5@example.com	Student	2025-05-17 01:25:35.133	
11	11	测试用户六	testuser6@example.com	Student	2025-05-17 01:25:35.133	
12	12	测试用户七	testuser7@example.com	Student	2025-05-17 01:25:35.133	

LibrarySeats					
	SeatID	ZoneID	SeatNumber	HasPowerOutlet	SeatStatus
1	F1Q-A01	1	A01	1	Available
2	F1Q-A02	1	A02	1	Available
3	F1Q-B01	1	B01	0	Maintenance
4	F3E-C01	3	C01	1	Available
5	F3E-C02	3	C02	1	Available
6	F3E-C03	3	C03	1	Maintenance
7	GSR201-S1	2	S1	1	Available
8	GSR201-S2	2	S2	1	Available

LibraryZones			结果		StaffAssignments				
	ZoneID	ZoneName	MaxBookingHours		AssignmentID	StaffID	ZoneID	AssignmentDate	Notes
1	1	安静学习区 一楼	3		1	4	1	2025-05-18 00:00:00.000	负责上午时段咨询
2	2	小组讨论室 201A	2		2	9	2	2025-05-18 00:00:00.000	负责下午设备维护
3	3	电子阅览区 三楼	4		3	4	4	2025-05-19 00:00:00.000	临时支援
4	4	休闲阅读角	1						

SeatReservations							
	ReservationID	UserID	SeatID	ReservationStartTime	ReservationEndTime	BookingTimestamp	ReservationStatus
1	1000	1	F1Q-A01	2025-05-17 09:00:00.000	2025-05-17 11:00:00.000	2025-05-17 01:25:35.150	Confirmed
2	1002	1	F3E-C01	2025-05-18 10:00:00.000	2025-05-18 12:00:00.000	2025-05-17 01:25:35.150	Confirmed
3	1001	2	GSR201-S1	2025-05-16 14:00:00.000	2025-05-16 15:00:00.000	2025-05-16 01:25:35.150	Completed
4	1003	3	F1Q-A02	2025-05-17 09:00:00.000	2025-05-17 10:00:00.000	2025-05-17 01:25:35.150	Confirmed
5	1004	6	F1Q-A02	2025-05-17 10:00:00.000	2025-05-17 12:00:00.000	2025-05-17 02:24:22.653	Confirmed
6	1005	7	GSR201-S1	2025-05-18 14:00:00.000	2025-05-18 15:30:00.000	2025-05-17 02:27:48.717	Confirmed
7	1006	8	F3E-C01	2025-05-19 09:00:00.000	2025-05-19 11:30:00.000	2025-05-17 02:31:00.460	Confirmed

5.2.1 实体完整性

(1) 列级定义主码（单属性构成的码）

```
-- LibraryUsers
UserID INT IDENTITY(1,1) PRIMARY KEY,
-- StaffAssignments
AssignmentID INT IDENTITY(1,1) PRIMARY KEY,
```

(2) 表级定义主码（单属性构成的码）

```
-- LibraryZones
CONSTRAINT PK_LibraryZones PRIMARY KEY (ZoneID);
-- SeatReservations
CONSTRAINT PK_LibrarySeats PRIMARY KEY (SeatID),
```

(3) 表级定义主码（复合主码）

```
CONSTRAINT PK_SeatReservations
PRIMARY KEY (UserID, SeatID, ReservationStartTime);
```

(4) 违反约束条件

`SeatReservations` 表中，`SeatID` 作为主码，当前已存在 `SeatID = 'F1Q-A01'` 的用户，再添加一条记录（`SeatID = 'F1Q-A01'`）会报错。

```
INSERT INTO LibrarySeats(SeatID, ZoneID, SeatNumber, HasPowerOutlet, SeatStatus)
VALUES ('F1Q-A01', 1, 'A0X', 0, 'Available');
```

```
INSERT INTO LibrarySeats (SeatID, ZoneID, SeatNumber, HasPowerOutlet, SeatStatus)
VALUES ('F1Q-A01', 1, 'A0X', 0, 'Available');
```

消息

消息 2627, 级别 14, 状态 1, 第 1 行
违反了 PRIMARY KEY 约束“PK_LibrarySeats”。不能在对象“dbo.LibrarySeats”中插入重复键。重复键值为 (F1Q-A01)。
语句已终止。

5.2.2 参照完整性

(1) 在表级定义参照完整性，显式说明违约处理（拒绝删除/级联更新）

```
-- LibrarySeats
CONSTRAINT FK_Seat_Zone FOREIGN KEY (ZoneID)
REFERENCES LibraryZones(ZoneID)
ON DELETE NO ACTION
ON UPDATE CASCADE,

-- SeatReservations
```

```

CONSTRAINT FK_Reservation_Seat FOREIGN KEY (SeatID)
    REFERENCES LibrarySeats(SeatID)
    ON DELETE NO ACTION
    ON UPDATE CASCADE,

-- StaffAssignments
CONSTRAINT FK_Assignment_Zone FOREIGN KEY (ZoneID)
    REFERENCES LibraryZones(ZoneID)
    ON DELETE NO ACTION
    ON UPDATE CASCADE,

```

这里我给出了三个示例，仅解释第一个（另外两个同理）

- `LibrarySeats.ZoneID` 参照 `LibraryZones.ZoneID`
- 被参照的 `LibraryZones.ZoneID` 禁止删除
- 更新 `LibraryZones.ZoneID` 时，`LibrarySeats` 中对应的 `ZoneID` 级联更新

(2) 在表级定义参照完整性，显式说明违约处理（级联删除/级联更新）

```

-- SeatReservations
CONSTRAINT FK_Reservation_User FOREIGN KEY (UserID)
    REFERENCES LibraryUsers(UserID)
    ON DELETE CASCADE
    ON UPDATE CASCADE,

-- StaffAssignments
CONSTRAINT FK_Assignment_Staff FOREIGN KEY (StaffID)
    REFERENCES LibraryUsers(UserID)
    ON DELETE SET NULL
    ON UPDATE CASCADE,

```

这里我给出了两个示例，仅解释第一个（另一个同理）

- `SeatReservations.UserID` 参照 `LibraryUsers.UserID`
- 删除 `LibraryUsers` 中的用户时，该用户在 `SeatReservations` 中的所有预订记录将级联删除
- 更新 `LibraryUsers.UserID` 主键值时，`SeatReservations` 中对应的 `UserID` 外键值也将级联更新

(3) 违反约束条件

插入时外键值在父表中不存在（`ZoneID=999` 不存在）

```

INSERT INTO LibrarySeats (SeatID, ZoneID, SeatNumber, SeatStatus)
VALUES ('TEST01', 999, 'T01', 'Available');

```

```
INSERT INTO LibrarySeats (SeatID, ZoneID, SeatNumber, SeatStatus)
VALUES ('TEST01', 999, 'T01', 'Available');
```

110 %

消息

消息 547, 级别 16, 状态 0, 第 1 行
INSERT 语句与 FOREIGN KEY 约束"FK_Seat_Zone"冲突。该冲突发生于数据库"LibraryDatabase", 表"dbo.LibraryZones", column 'ZoneID'。
语句已终止。

(4) 删除被参照的父表记录 (ON DELETE NO ACTION)

```
DELETE FROM LibraryZones WHERE ZoneID = 1;
```

阻止删除

```
DELETE FROM LibraryZones WHERE ZoneID = 1;
```

0 %

消息

消息 547, 级别 16, 状态 0, 第 1 行
DELETE 语句与 REFERENCE 约束"FK_Seat_Zone"冲突。该冲突发生于数据库"LibraryDatabase", 表"dbo.LibrarySeats", column 'ZoneID'。
语句已终止。

(5) 删除被参照的父表记录 (ON DELETE CASCADE)

```
DELETE FROM LibraryUsers WHERE UserID = 1;
```

删除前后查询, 发现 **SeatReservations** 的记录连同 **LibraryUsers** 的记录一起被删除

```
SELECT * FROM SeatReservations WHERE UserID = 1;
DELETE FROM LibraryUsers WHERE UserID = 1;
SELECT * FROM SeatReservations WHERE UserID = 1;
SELECT * FROM LibraryUsers WHERE UserID = 1;
```

110 %

结果 消息

	ReservationID	UserID	SeatID	ReservationStartTime	ReservationEndTime	BookingTimestamp	ReservationStatus
1	1000	1	F1Q-A01	2025-05-17 09:00:00.000	2025-05-17 11:00:00.000	2025-05-17 01:25:35.150	Confirmed
2	1002	1	F3E-C01	2025-05-18 10:00:00.000	2025-05-18 12:00:00.000	2025-05-17 01:25:35.150	Confirmed

ReservationID	UserID	SeatID	ReservationStartTime	ReservationEndTime	BookingTimestamp	ReservationStatus
---------------	--------	--------	----------------------	--------------------	------------------	-------------------

UserID	FullName	Email	UserType	RegistrationDate
--------	----------	-------	----------	------------------

(6) 更新父表主键 (ON UPDATE CASCADE)

```
UPDATE LibrarySeats SET SeatID = 'NEW-S02' WHERE SeatID = 'F1Q-A02';
```

更新前后查询，发现 `SeatReservations` 的记录连同 `LibrarySeats` 的记录一起更新

```
SELECT ReservationID, UserID, SeatID, ReservationStartTime, ReservationStatus
FROM SeatReservations WHERE SeatID = 'F1Q-A02';
SELECT * FROM LibrarySeats WHERE SeatID = 'F1Q-A02';
UPDATE LibrarySeats SET SeatID = 'NEW-S02' WHERE SeatID = 'F1Q-A02';
SELECT ReservationID, UserID, SeatID, ReservationStartTime, ReservationStatus
FROM SeatReservations WHERE SeatID = 'NEW-S02';
SELECT ReservationID, UserID, SeatID, ReservationStartTime, ReservationStatus
FROM SeatReservations WHERE SeatID = 'F1Q-A02';
SELECT * FROM LibrarySeats WHERE SeatID = 'NEW-S02';
```

110 %

结果 消息

	ReservationID	UserID	SeatID	ReservationStartTime	ReservationStatus
1	1003	3	F1Q-A02	2025-05-17 09:00:00.000	Confirmed
2	1004	6	F1Q-A02	2025-05-17 10:00:00.000	Confirmed

	SeatID	ZoneID	SeatNumber	HasPowerOutlet	SeatStatus
1	F1Q-A02	1	A02	1	Available

更新前

	ReservationID	UserID	SeatID	ReservationStartTime	ReservationStatus
1	1003	3	NEW-S02	2025-05-17 09:00:00.000	Confirmed
2	1004	6	NEW-S02	2025-05-17 10:00:00.000	Confirmed

ReservationID	UserID	SeatID	ReservationStartTime	ReservationStatus

	SeatID	ZoneID	SeatNumber	HasPowerOutlet	SeatStatus
1	NEW-S02	1	A02	1	Available

更新后

5.2.3 用户定义完整性

(1) NULL（允许为空值）/NOT NULL（不允许为空值）

代码中有很多，这里仅挑几个示例

```
-- LibraryUsers
-- 使用CHECK实现NOT NULL效果(表级)
CONSTRAINT CK_FullName_NotNull CHECK (FullName IS NOT NULL);

-- LibrarySeats(列级)
ZoneID INT NOT NULL,
SeatNumber VARCHAR(5) NULL,
```

(2) UNIQUE（列值唯一）

代码中有很多，这里仅挑几个示例

```
-- LibraryZones(列级)
```



```

ZoneName NVARCHAR(50) NOT NULL UNIQUE,

-- SeatReservations(表级)
CONSTRAINT UQ_ReservationID UNIQUE (ReservationID),

-- StaffAssignments(复合)
CONSTRAINT UQ_Staff_Zone_Day
    UNIQUE (StaffID, ZoneID, AssignmentDate)

```

(3) 违反约束条件

违反 NOT NULL

```

INSERT INTO LibrarySeats(SeatID, ZoneID, SeatNumber, HasPowerOutlet, SeatStatus)
VALUES ('F3E-C04', NULL, 'C04', 0, 'Available');

```

```

INSERT INTO LibrarySeats (SeatID, ZoneID, SeatNumber, HasPowerOutlet, SeatStatus)
VALUES ('F3E-C04', NULL, 'C04', 0, 'Available');

```

10 %

消息

消息 515, 级别 16, 状态 2, 第 1 行
不能将值 NULL 插入列 'ZoneID', 表 'LibraryDatabase.dbo.LibrarySeats'; 列不允许有 Null 值。INSERT 失败。
语句已终止。

违反 UNIQUE

```

INSERT INTO LibraryZones (ZoneName, MaxBookingHours)
VALUES (N'安静学习区 一楼', 2);

```

```

INSERT INTO LibraryZones (ZoneName, MaxBookingHours)
VALUES (N'安静学习区 一楼', 2); -- 尝试插入一个已存在的 ZoneName

```

110 %

消息

消息 2627, 级别 14, 状态 1, 第 1 行
违反了 UNIQUE KEY 约束"UQ_ZoneName"。不能在对象"dbo.LibraryZones"中插入重复键。重复键值为 (安静学习区 一楼)。
语句已终止。

5.2.4 CHECK短语

(1) 指定列值应该满足的条件 (IN/BETWEEN)

```

-- LibraryUsers
CONSTRAINT CK_UserType
    CHECK (UserType IN ('Student', 'Faculty', 'Staff'));

-- LibrarySeats
CHECK (SeatStatus IN ('Available', 'Maintenance'));

-- LibraryZones
CONSTRAINT CK_ZoneMaxHours

```



```
CHECK (MaxBookingHours BETWEEN 1 AND 4);
```

(2) 设置元组不同属性间的约束条件

```
-- SeatReservations
CONSTRAINT CK_ReservationTimesValid
    CHECK (ReservationEndTime > ReservationStartTime);

CONSTRAINT CK_BookingTimestamp
    CHECK (BookingTimestamp <= ReservationStartTime)
```

(3) 违反约束条件

UserType 取了 Student, Faculty, Staff 之外的 Visitor

```
INSERT INTO LibraryUsers (FullName, Email, UserType, RegistrationDate)
VALUES ('Test ', 'visitor.test@example.com', 'Visitor', GETDATE());
```



5.2.5 CONSTRAINT子句

(1) CONSTRAINT 约束

建库代码中含有大量此类约束语句

```
-- LibraryUsers
CONSTRAINT CK_UserType
    CHECK (UserType IN ('Student', 'Faculty', 'Staff'))

-- SeatReservations
CONSTRAINT UQ_ReservationID UNIQUE (ReservationID),
```

(2) 违反约束条件

违反 CONSTRAINT CK_UserType CHECK (UserType IN ('Student', 'Faculty', 'Staff'))

```
INSERT INTO LibraryUsers (FullName, Email, UserType, RegistrationDate)
VALUES ('Test ', 'visitor.test@example.com', 'Visitor', GETDATE());
```

可以看到报错中使用的命名正是 `CK_UserType`

```
INSERT INTO LibraryUsers (FullName, Email, UserType, RegistrationDate)
VALUES ('Test ', 'visitor.test@example.com', 'Visitor', GETDATE());
```

110 %

消息

消息 547, 级别 16, 状态 0, 第 1 行
INSERT 语句与 CHECK 约束"CK_UserType"冲突。该冲突发生于数据库"LibraryDatabase", 表"dbo.LibraryUsers", column 'UserType'。
语句已终止。

(3) ALTER DROP/ALTER ADD

对以下语句进行处理，首先验证其有效性

```
-- LibraryZones
CONSTRAINT CK_ZoneMaxHours CHECK (MaxBookingHours BETWEEN 1 AND 4)
```

插入一条违反约束的语句，证明约束有效

```
INSERT INTO LibraryZones (ZoneName, MaxBookingHours)
VALUES (N'超大预订区 - 测试原约束', 5);
```

```
INSERT INTO LibraryZones (ZoneName, MaxBookingHours)
VALUES (N'超大预订区 - 测试原约束', 5);
```

10 %

消息

消息 547, 级别 16, 状态 0, 第 1 行
INSERT 语句与 CHECK 约束"CK_ZoneMaxHours"冲突。该冲突发生于数据库"LibraryDatabase", 表"dbo.LibraryZones", column 'MaxBookingHours'。
语句已终止。

删除约束

```
ALTER TABLE LibraryZones
DROP CONSTRAINT CK_ZoneMaxHours;
```

再插入语句，插入成功

```
INSERT INTO LibraryZones (ZoneName, MaxBookingHours)
VALUES (N'超大预订区 - 测试原约束', 6);
SELECT * FROM LibraryZones;
```

110 %

结果 消息

	ZoneID	ZoneName	MaxBookingHours
1	1	安静学习区 一楼	3
2	2	小组讨论室 201A	2
3	3	电子阅览区 三楼	4
4	4	休闲阅读角	1
5	8	超大预订区 - 测试原约束	6

添加约束（先删除表中已有不满足该约束的）

```
DELETE FROM LibraryZones WHERE MaxBookingHours > 5;
ALTER TABLE LibraryZones ADD CONSTRAINT CK_ZoneMaxHours_New
CHECK (MaxBookingHours BETWEEN 1 AND 5);
SELECT * FROM LibraryZones;
```

110 %

结果 消息

	ZoneID	ZoneName	MaxBookingHours
1	1	安静学习区 一楼	3
2	2	小组讨论室 201A	2
3	3	电子阅览区 三楼	4
4	4	休闲阅读角	1

验证新约束有效性

```
INSERT INTO LibraryZones (ZoneName, MaxBookingHours)
VALUES (N'超大预订区 - 测试新约束', 6);
```

10 %

消息

消息 547, 级别 16, 状态 0, 第 1 行
INSERT 语句与 CHECK 约束"CK_ZoneMaxHours_New"冲突。该冲突发生于数据库"LibraryDatabase", 表"dbo.LibraryZones", column 'MaxBookingHours'。
语句已终止。

6. 问题与解决

问题：使用创建的登录名登录时，弹出错误

解决：勾选"信任服务器证书"

