

# 功能简介和数据库部分实现

---

18308045 Zhengyang Gu

January 15, 2021

## Contents

<b>1</b>	<b>功能简介</b>	<b>2</b>
<b>2</b>	<b>具体实现</b>	<b>2</b>
2.1	模式设计 . . . . .	2
2.1.1	ER 图 . . . . .	2
2.1.2	转化为模式 . . . . .	2
2.2	触发器 . . . . .	4
2.2.1	评论 . . . . .	4
2.2.2	交易 . . . . .	5
2.3	可能出现的不一致问题 . . . . .	6

## 1 功能简介

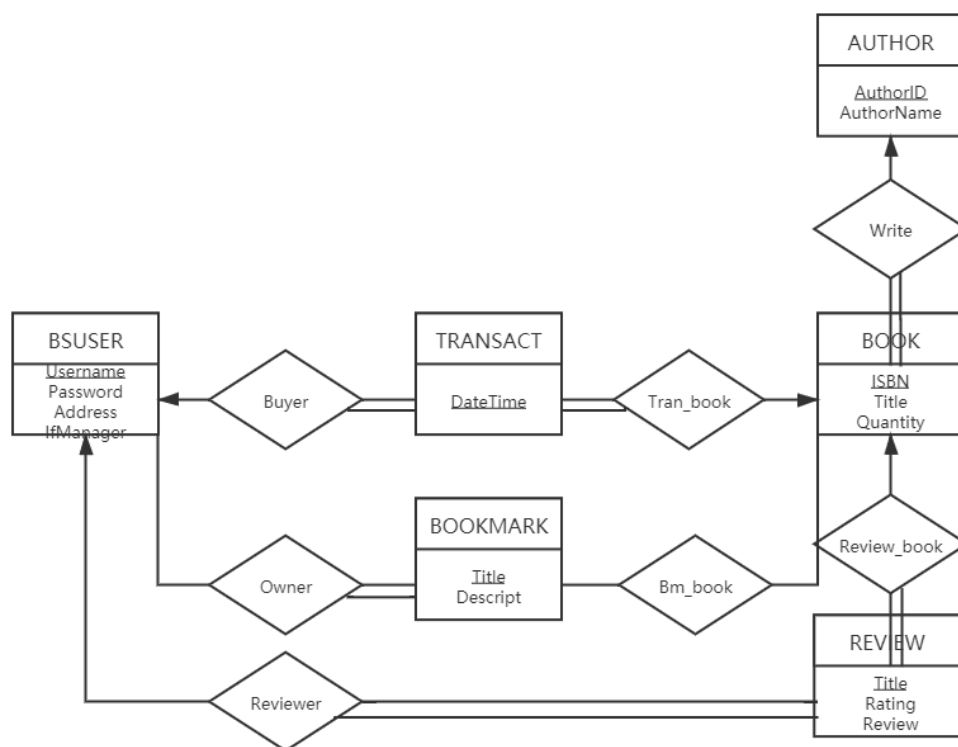
总体来说是图书售卖系统。具体来说，包含以下几个功能：

1. 创建管理员用户，管理员用户可以管理数据库。
2. 创建普通用户。
3. 如果图书还有余量，则可购买此图书。
4. 如果用户已经购买过图书，则可评价此图书。
5. 创建书单其中有标题和描述，所有用户都可以查看，并按照书单买书（好书推荐）。

## 2 具体实现

### 2.1 模式设计

#### 2.1.1 ER 图



#### 2.1.2 转化为模式

一对多的，让多的实体包含一的实体的主码；多对多的，增加一个表包含两个实体。另外值得注意的是对书的数目即 quantity 设了一个约束大于等于 0。

```
1 CREATE TABLE BSUSER
```

```

2  (
3      Username VARCHAR(15) NOT NULL,
4      Password VARCHAR(15) NOT NULL,
5      Address VARCHAR(20) NOT NULL,
6      IfManager INT,
7      PRIMARY KEY(Username)
8  );
9
10 CREATE TABLE AUTHOR
11 (
12     AuthorID VARCHAR(15) NOT NULL,
13     AuthorName VARCHAR(15) NOT NULL,
14     PRIMARY KEY(AuthorID)
15 );
16
17 CREATE TABLE BOOK
18 (
19     ISBN CHAR(14) NOT NULL,
20     Title VARCHAR(50) NOT NULL,
21     AuthorID VARCHAR(15) NOT NULL,
22     Quantity INT NOT NULL,
23     CHECK (Quantity >= 0),
24     PRIMARY KEY(ISBN),
25     FOREIGN KEY(AuthorID) references AUTHOR(AuthorID)
26 );
27
28 CREATE TABLE REVIEW
29 (
30     Title VARCHAR(15),
31     Reviewer VARCHAR(15) NOT NULL,
32     BookID CHAR(14) NOT NULL,
33     Rating INT NOT NULL,
34     Review TEXT,
35     CHECK (Rating >= 1 OR Rating <= 5),
36     PRIMARY KEY(Title),
37     FOREIGN KEY(Reviewer) references BSUSER(Username),
38     FOREIGN KEY(BookID) references BOOK(ISBN)
39 );
40
41 CREATE TABLE TRANSACT
42 (

```

```

43     DateTime CHAR(23) NOT NULL,
44     Buyer VARCHAR(15) NOT NULL,
45     BookID CHAR(14) NOT NULL,
46     PRIMARY KEY(DateTime),
47     FOREIGN KEY(Buyer) references BSUSER(Username),
48     FOREIGN KEY(BookID) references BOOK(ISBN)
49 );
50
51 create table BOOKMARK
52 (
53     Title VARCHAR(15),
54     Descript TEXT,
55     primary key(Title)
56 )
57
58 create table BOOKMARK_BOOK
59 (
60     Title VARCHAR(15),
61     Owner VARCHAR(15),
62     BookID CHAR(14),
63     Owned Bit,
64     primary key(Title, BookID, Owned),
65     foreign key(Owner) references BSUSER(Username),
66     foreign key(BookID) references BOOK(ISBN)
67 )

```

## 2.2 触发器

### 2.2.1 评论

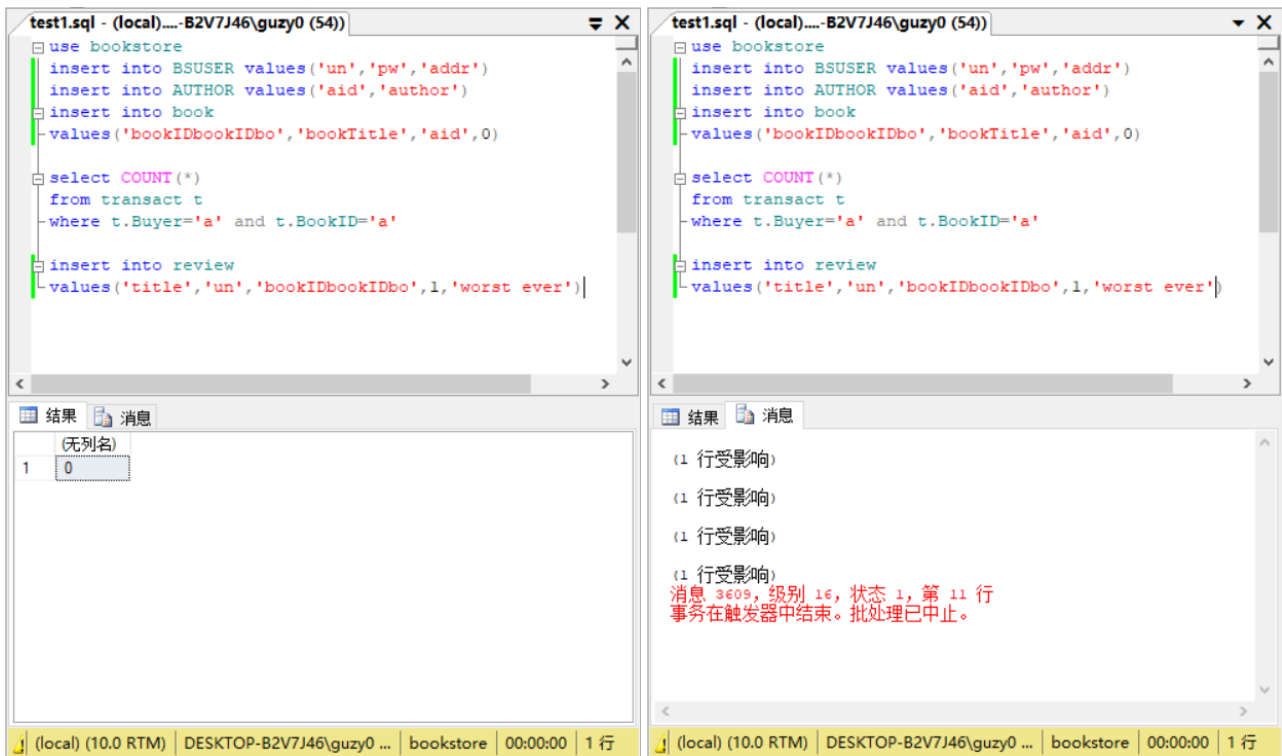
要求评论书前，需要已经购买过此书：

```

1 create trigger t_reivew on REVIEW for insert as
2 if (select count(*)
3     from TRANSACT t, inserted i
4     where t.Buyer=iReviewer and t.BookID=i.BookID)=0
5 begin
6     rollback TRANSACTION
7 end

```

测试恶意评论结果如下：



可以看出，用户'un' 对书'bookIDbookIDbo' 的交易为 0，触发器致使'un' 评论失败。

## 2.2.2 交易

购买书的时候，有以下几点：

1. 书的数量要大于 0。
2. 买完后书的数量减 1。
3. 书单上标记已购此书。

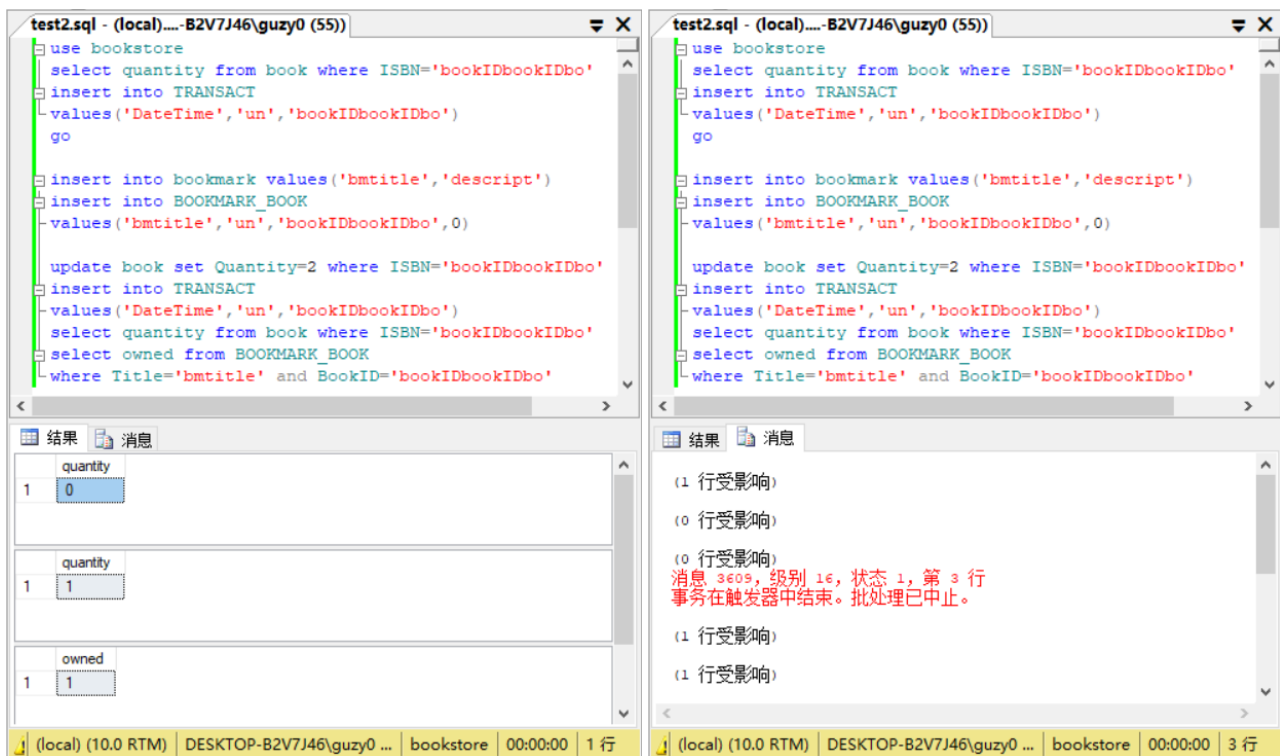
实现如下：

```

1 create trigger t_transact on TRANSACT for insert as
2 if (select quantity from book, inserted where ISBN=inserted.BookID) <= 0
3 begin
4     rollback transaction
5 end
6 update book set quantity=quantity-1 where ISBN=(select BookID from inserted)
7 update bookmark_book set Owned=1 where BookID=(select BookID from inserted)
8 and Owner=(select Buyer from inserted)

```

测试如下：



可以看出，书的数量是 0，触发器致使'un' 交易失败。设置此书于书单'bmtitle' 中且标记为无此书，更新书的数量为 2。之后再插入交易，发现成功，且交易后书的数量变为 1，书单中标记为拥有此书。

## 2.3 可能出现的不一致问题

由于触发器并不是原子操作，在多用户多事务并发的环境下插入交易时可能出现不一致的情况。考虑此时某本书只有一个，事务 A 在刚刚判断完  $quantity > 0$  而未修改  $quantity$  并提交，此时事务 B 也判断会认为  $quantity > 0$ ，最终结果是一本书插入了两条交易。下面在触发器判断后和插入前加一个 10 秒延迟来模拟这种现象。test3.sql 代码截图不全，因而在此放出：

```

1 use bookstore
2 drop trigger t_transact
3 go
4 create trigger t_transact on TRANSACT for insert as
5 if (select quantity from book, inserted where ISBN=inserted.BookID) <= 0
6 begin
7 rollback transaction
8 end
9 waitfor delay '00:00:10'
10 update book set quantity=quantity-1 where ISBN=(select BookID from inserted)
11 update bookmark_book set Owned=1 where BookID=(select BookID from inserted)

```

```

12 and Owner=(select Buyer from inserted)
13 go
14
15 update book set quantity=1 where ISBN='bookIDbookIDbo'
16 begin tran
17 set transaction isolation level read committed
18 select quantity from book where ISBN='bookIDbookIDbo'
19 insert into TRANSACT
20 values('DateTime1','un','bookIDbookIDbo')
21 commit tran

```

其主要功能就是在 t\_transact 的判断语句和插入语句之间加一个延迟，将书 quantity 设为 1 并执行事务插入一条交易。运行结果如下：

The image shows two side-by-side screenshots of the SQL Server Enterprise Manager interface, specifically the 'test3.sql' file. Both windows show the same SQL script, which includes a 10-second delay before the update statement. The left window shows the script with the delay, and the right window shows the script with the delay. Below the script, the 'Results' pane shows the outcome of the transaction. In the left window, the 'quantity' column is updated to 1. In the right window, the 'Results' pane shows five rows of '1 行受影响' (1 row affected).

quantity
1

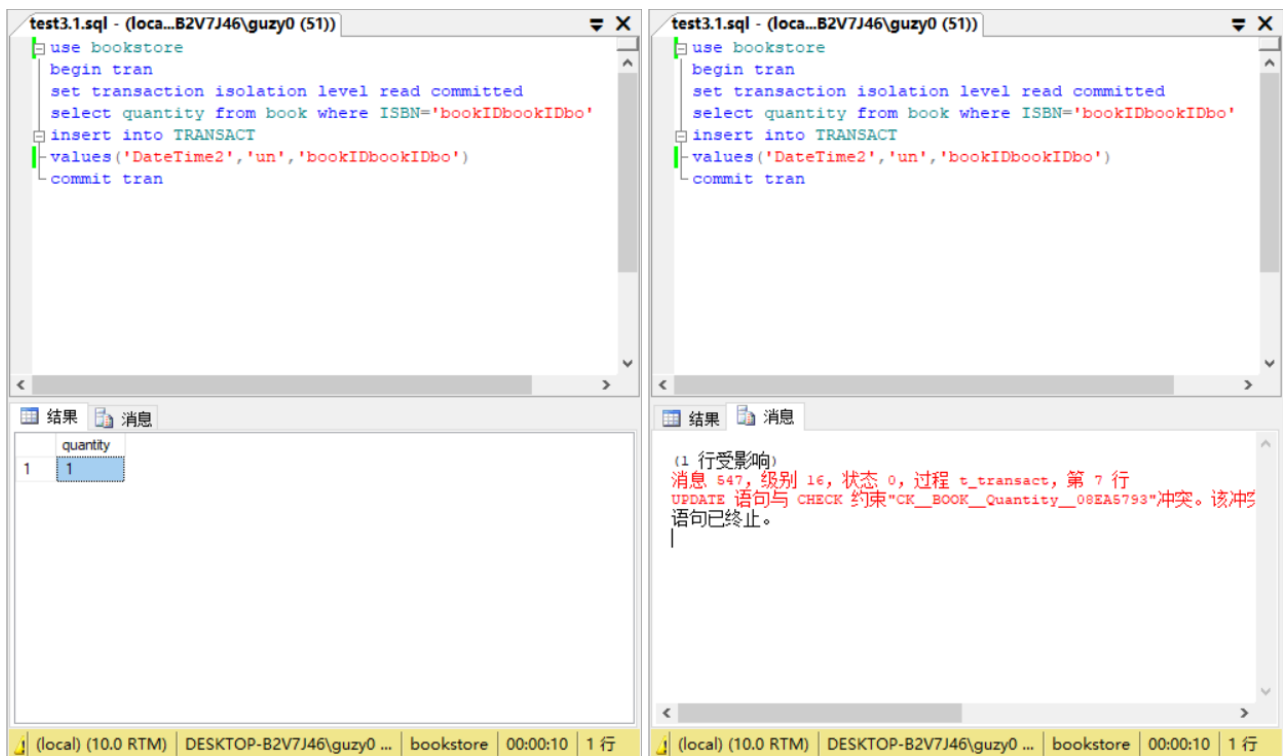
Results (Left Window):

quantity
1

Results (Right Window):

(1 行受影响)
(1 行受影响)
(1 行受影响)
(1 行受影响)
(1 行受影响)

上面事务执行的同时，执行如下事务：



可以发现，两个事务都读到书的 quantity 为 1，因此两个事务都没被回滚，执行了插入语句。但是由于 quantity 的约束要求大于等于 0，其中一个事务的 trigger 更新会失败，导致 trigger 失败进而导致插入语句失败。因此 update 语句和对 quantity 的约束已经防止了此类不一致的发生，因而提前判断 quantity 也是没必要的，因而改成：

```

1 create trigger t_transact on TRANSACT for insert as
2 update book set quantity=quantity-1 where ISBN=(select BookID from inserted)
3 update bookmark_book set Owned=1 where BookID=(select BookID from inserted)
4 and Owner=(select Buyer from inserted)

```