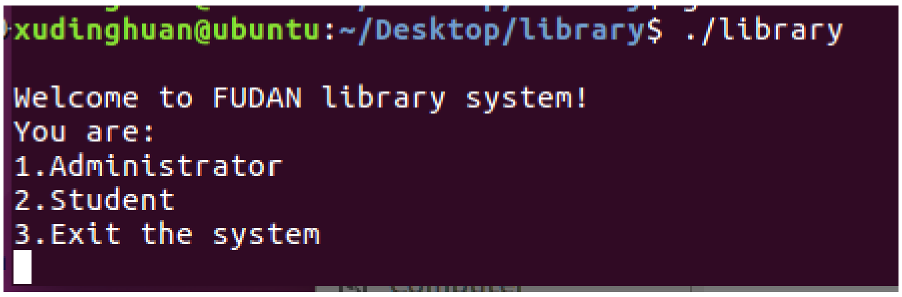
图书管理系统设计报告

实验人：徐丁焕
学号：18307130031
完成日期：2020-5-3

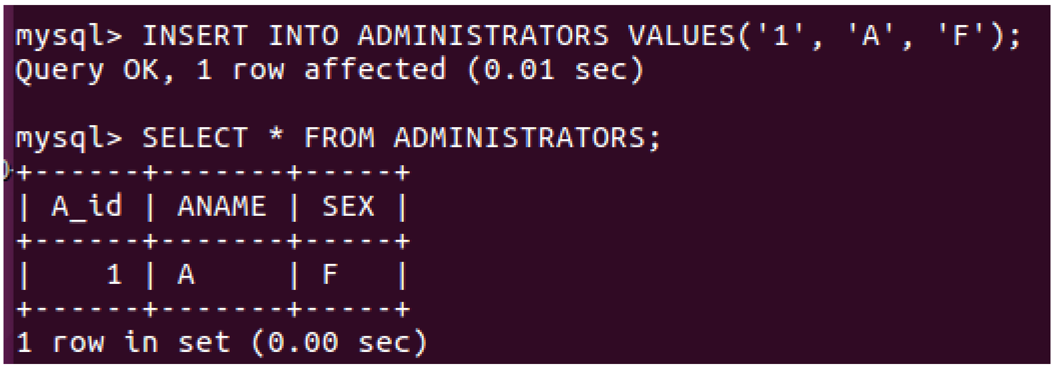
一．系统交互界面：

本人实现了一个比较简单的 command line interface,在执行.EXE 文件后出现：



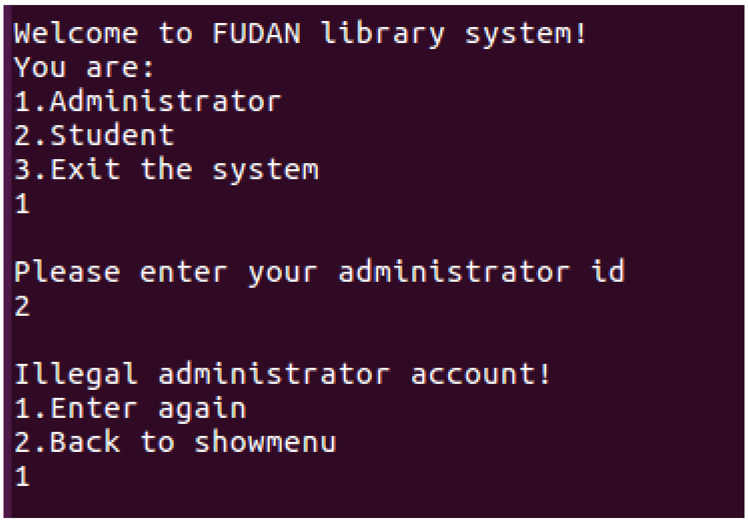
```
xudinghuan@ubuntu:~/Desktop/library$ ./library

Welcome to FUDAN library system!
You are:
1.Administrator
2.Student
3.Exit the system
```

通过输入左侧数字与系统进行互动。因为从所要求实现的功能来看，没有人有创建管理员账户的能力，为了能够进行操作，假设已经存在序号为 1 的一名女性管理员 A。（有哪些表及属性的分配会在第二部分阐述，第一部分仅是演示交互界面）



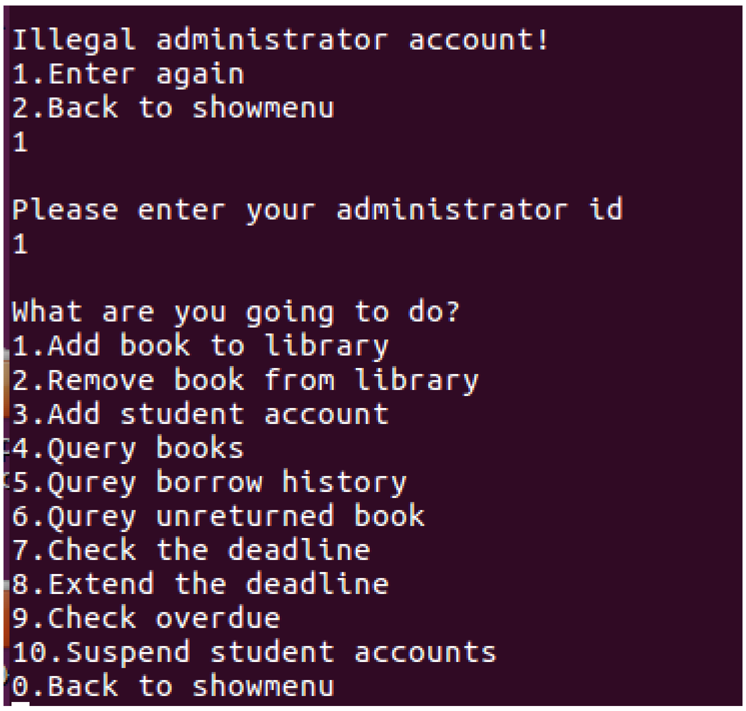
```
mysql> INSERT INTO ADMINISTRATORS VALUES('1', 'A', 'F');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM ADMINISTRATORS;
+------+-------+-----+
| A_id | ANAME | SEX |
+------+-------+-----+
|    1 | A     | F   |
+------+-------+-----+
1 row in set (0.00 sec)
```

试图以管理员的身份登录，但输入了不存在的管理员 id，此时有一个纠错机制：



```
Welcome to FUDAN library system!
You are:
1.Administrator
2.Student
3.Exit the system
1

Please enter your administrator id
2

Illegal administrator account!
1.Enter again
2.Back to showmenu
1
```

在输入了正确的管理员 id 后，显示出管理员功能菜单：（学生的登入模式同理）



```
Illegal administrator account!
1.Enter again
2.Back to showmenu
1

Please enter your administrator id
1

What are you going to do?
1.Add book to library
2.Remove book from library
3.Add student account
4.Query books
5.Qurey borrow history
6.Qurey unreturned book
7.Check the deadline
8.Extend the deadline
9.Check overdue
10.Suspend student accounts
0.Back to showmenu
```

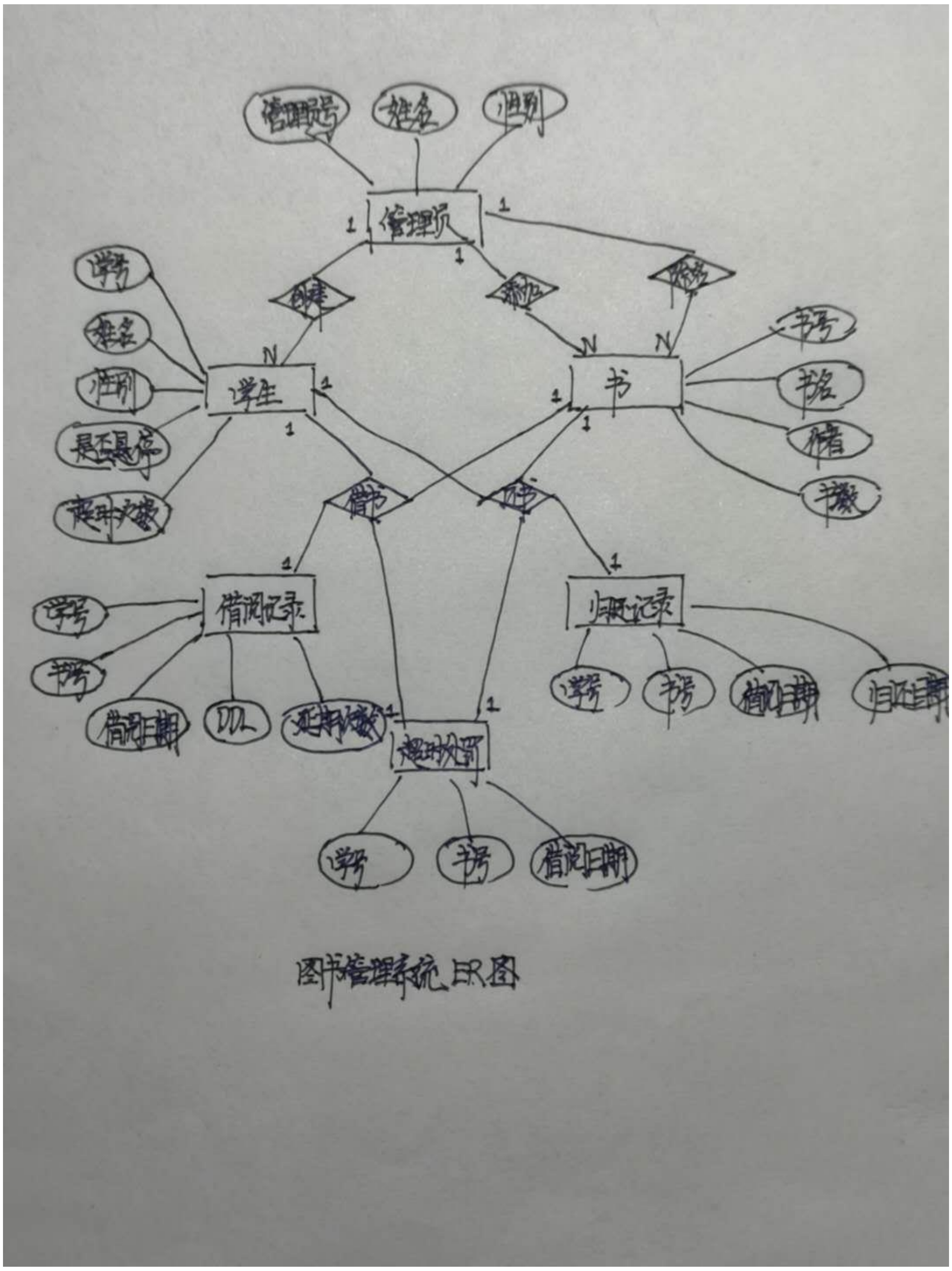
指定任务也是通过输入左侧的数字，而当每次完成一次指定的任务后（任务自然是通过函数完成，各函数放在第三部分逐一阐述），会重新返回这个管理员菜单界面，直到输入 0 后返回主界面。

另外，值得一提的是，虽然在项目指导中规定了需要实现哪些功能，但没有具体规定管理者和学生是不是都有权限使用这些功能，于是便按照自己对图书系统的理解分配了。管理员的授权功能如上图所示，几乎可以使用所有功能，但显然管理员是不需要借书和还书的，所以一共 10 个功能。
对于学生而言，学生可以借书和还书，但其它的功能的使用应该受限：不能将书添加到图书馆中或是从图书馆中将某本书除名；不能创建其它学生的账号；只能查询本人的各种信息；不能直接延长 deadline(需要经手管理员操作)。

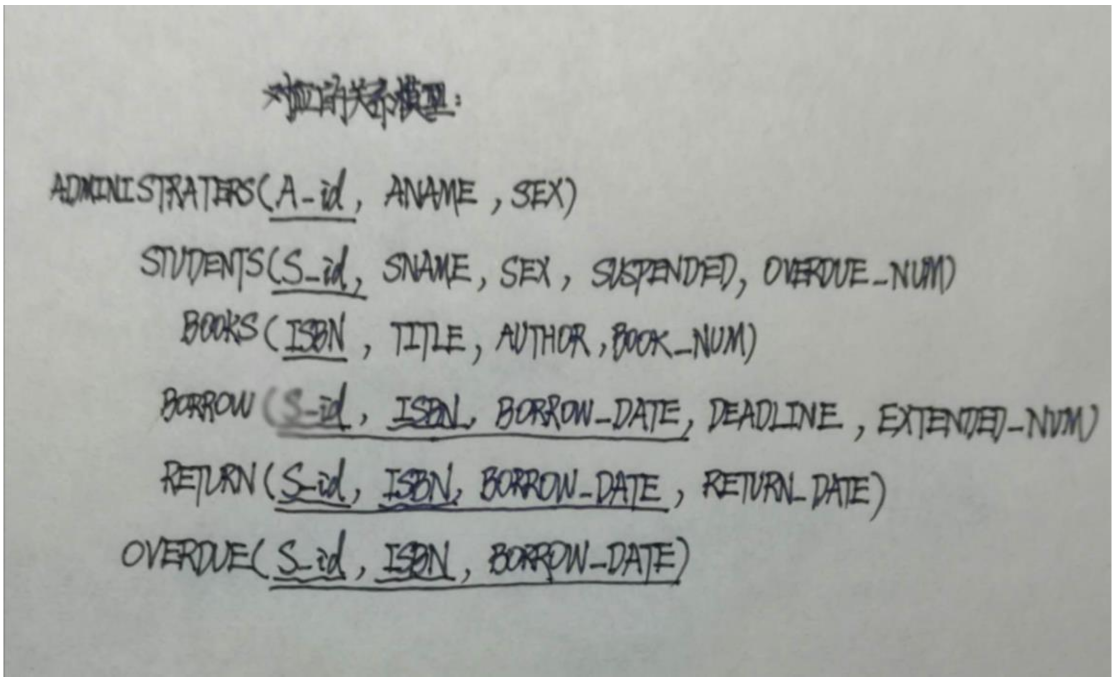
二．ER 图与关系模型：

根据课堂上讲授的理论知识，设计的第一步便应该是在分析用户需求的基础上画出 ER 图并将之转换成关系模型以建表。

ER 图如下：
共计 6 个实体：管理员、学生、书、借阅记录、归还记录和超时处罚。
各主体的主、外键（同上方顺序）：管理员号、学号、书号。由于同一种书很有可能被不同的学生于不同的时间点借走，因此后三者的主键都是（学号，书号，借阅日期），显然其中的学号和书号都是对应实体的外键。


管理员编号
姓名
性别
管理员
1
1
1
创建
添加
维护
N
N
N
学号
姓名
性别
是否是信
超时次数
学生
1
1
书
1
1
书号
书名
作者
书数
借书
还书
1
1
借阅记录
归还记录
学号
书号
借阅日期
DDL
延期次数
学号
书号
借阅日期
归还日期
1
1
超时处罚
学号
书号
借阅日期
图书管理系统 ER图

所对应的关系模型如下:



对应的关系模型:

ADMINISTRATERS(A-id, ANAME, SEX)

STUDENTS(S_id, SNAME, SEX, SUSPENDED, OVERDUE_NUM)

BOOKS(ISBN, TITLE, AUTHOR, BOOK_NUM)

BORROW(S-id, ISBN, BORROW-DATE, DEADLINE, EXTENDED-NUM)

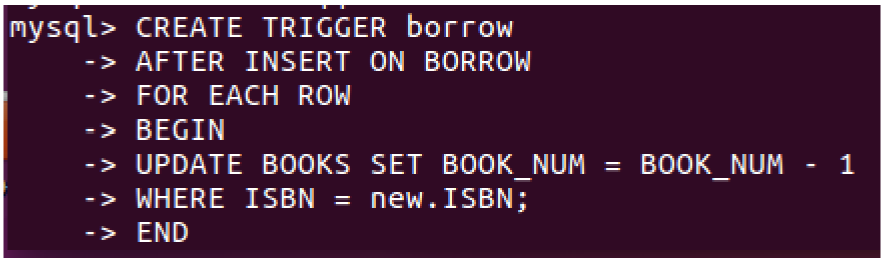
RETURN(S-id, ISBN, BORROW-DATE, RETURN-DATE)

OVERDUE(S-id, ISBN, BORROW-DATE)

为了便于实现一些功能，在这些表上设置一些了一些触发器（trigger）并在库（名为LIBRARY）中设置了一个定时器(event)：

A．触发器 borrow:

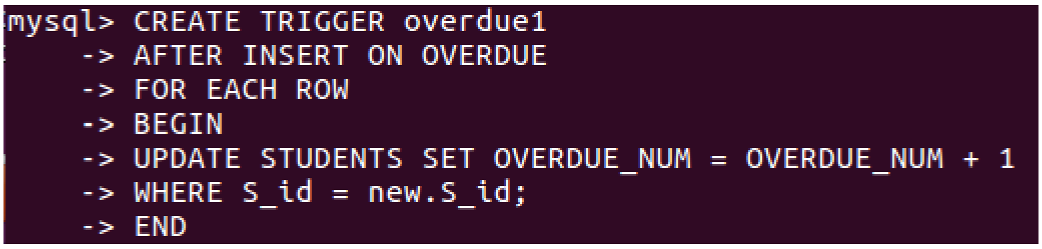
在有书被借走后（借阅记录被插入），相应的现有书数自减：



```
mysql> CREATE TRIGGER borrow
    -> AFTER INSERT ON BORROW
    -> FOR EACH ROW
    -> BEGIN
    -> UPDATE BOOKS SET BOOK_NUM = BOOK_NUM - 1
    -> WHERE ISBN = new.ISBN;
    -> END
```

B．触发器ｏｖｅｒｄｕｅ１：

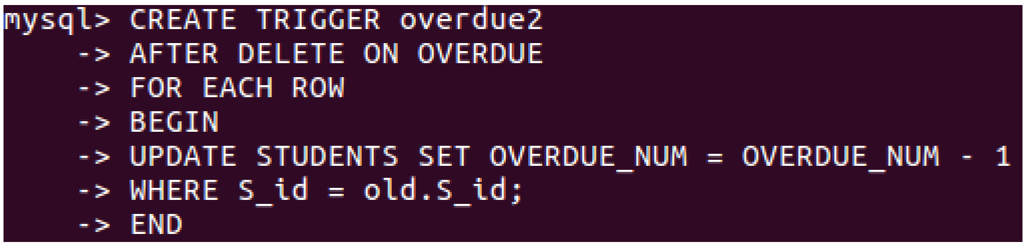
当有超时处罚单被生成后，对应学生的超时次数自增：



```
mysql> CREATE TRIGGER overdue1
    -> AFTER INSERT ON OVERDUE
    -> FOR EACH ROW
    -> BEGIN
    -> UPDATE STUDENTS SET OVERDUE_NUM = OVERDUE_NUM + 1
    -> WHERE S_id = new.S_id;
    -> END
```

C．触发器ｏｖｅｒｄｕｅ２：

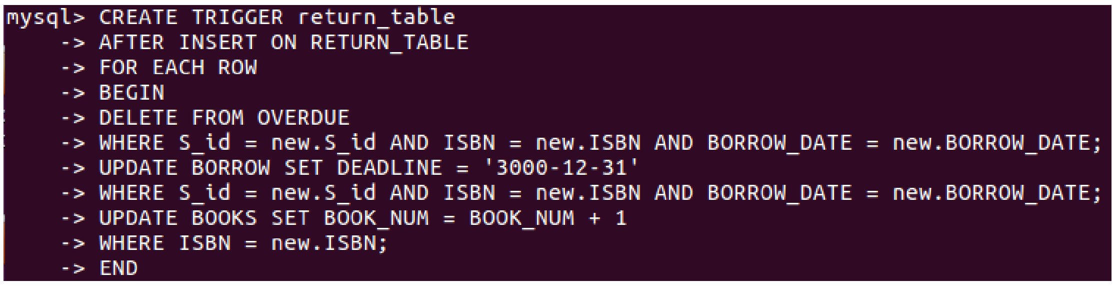
当有超时处罚单被撤销后，对应学生的超时次数自减：



```
mysql> CREATE TRIGGER overdue2
    -> AFTER DELETE ON OVERDUE
    -> FOR EACH ROW
    -> BEGIN
    -> UPDATE STUDENTS SET OVERDUE_NUM = OVERDUE_NUM - 1
    -> WHERE S_id = old.S_id;
    -> END
```

D．触发器ｒｅｔｕｒｎ＿ｔａｂｌｅ：

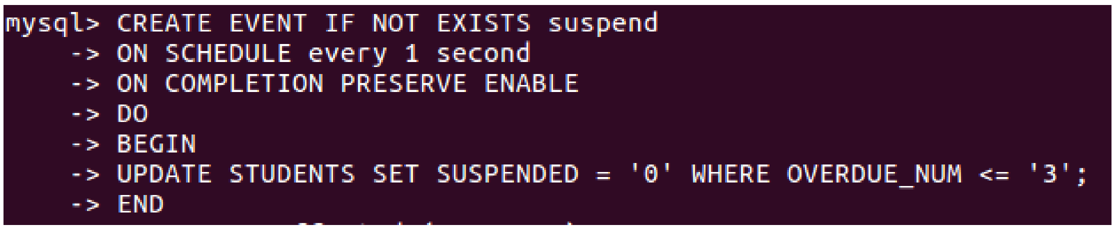
当有书被归还（归还记录被插入）时： １.撤销对应的超时处罚单。２.将对应的借阅记录中的ＤＥＡＤＬＩＮＥ改为一遥远日期３０００－１２－３１（便于后续的定时器判断）３.将对应书的现有书数自增：



```
mysql> CREATE TRIGGER return_table
    -> AFTER INSERT ON RETURN_TABLE
    -> FOR EACH ROW
    -> BEGIN
    -> DELETE FROM OVERDUE
    -> WHERE S_id = new.S_id AND ISBN = new.ISBN AND BORROW_DATE = new.BORROW_DATE;
    -> UPDATE BORROW SET DEADLINE = '3000-12-31'
    -> WHERE S_id = new.S_id AND ISBN = new.ISBN AND BORROW_DATE = new.BORROW_DATE;
    -> UPDATE BOOKS SET BOOK_NUM = BOOK_NUM + 1
    -> WHERE ISBN = new.ISBN;
    -> END
```

E．定时器ｓｕｓｐｅｎｄ：

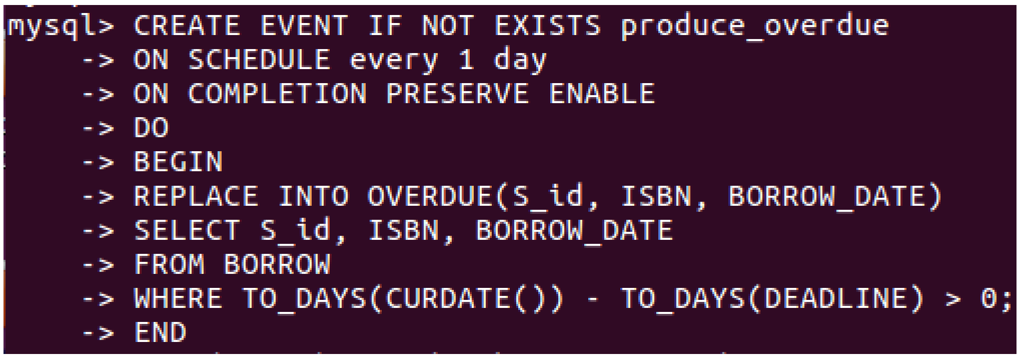
每秒确认学生的超时次数是否已经＜＝３，从而可以解除学生的悬停状态：



```
mysql> CREATE EVENT IF NOT EXISTS suspend
    -> ON SCHEDULE every 1 second
    -> ON COMPLETION PRESERVE ENABLE
    -> DO
    -> BEGIN
    -> UPDATE STUDENTS SET SUSPENDED = '0' WHERE OVERDUE_NUM <= '3';
    -> END
```

F．定时器 produce_overdue：

该定时器每隔一天（存储的各种时间信息都是ＤＡＴＥ类型）查看借阅记录中的ＤＥＡＤＬＩＮＥ有没有超过系统当前日期的（如果在ｄｄｌ之前已经还了书，之前设置的触发器会把ｄｄｌ改为极遥远的日期，保证不会触发定时器），若有则生成对应的超时处罚单：



```
mysql> CREATE EVENT IF NOT EXISTS produce_overdue
    -> ON SCHEDULE every 1 day
    -> ON COMPLETION PRESERVE ENABLE
    -> DO
    -> BEGIN
    -> REPLACE INTO OVERDUE(S_id, ISBN, BORROW_DATE)
    -> SELECT S_id, ISBN, BORROW_DATE
    -> FROM BORROW
    -> WHERE TO_DAYS(CURDATE()) - TO_DAYS(DEADLINE) > 0;
    -> END
```

有了这些触发器和定时器之后，在各个Ｇｏ函数中只需要做很少的事情就可以完成系统的功能。

三．功能函数的实现：

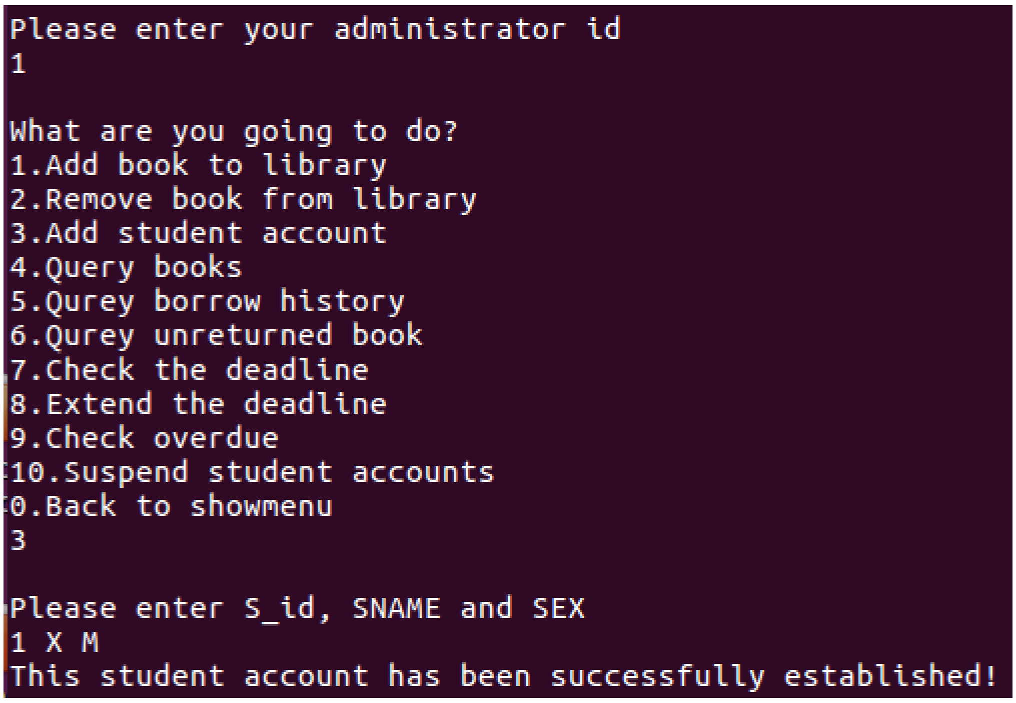
１．创建学生账户的ＡｄｄＡｃｃｏｕｎｔ函数：

```
func AddAccount(S_id int, SNAME string, SEX string){
        db, err := sql.Open("mysql", "root:xudinghuan@tcp(127.0.0.1:3306)/LIBRARY")

        if err != nil{
                fmt.Printf("Open database error!\n")
                return
        }

        defer db.Close()

        _, err = db.Exec("INSERT INTO STUDENTS(S_id, SNAME, SEX, SUSPENDED, OVERDUE_NUM)VALUES (?,?,?,0,0)",
S_id, SNAME, SEX)
        if err != nil{
                fmt.Printf("Add student account error!\n")
                return
        }
        return
}
```

一个简单的插入语句就能实现，注意到一个新创建的学生用户后两个字段肯定是０

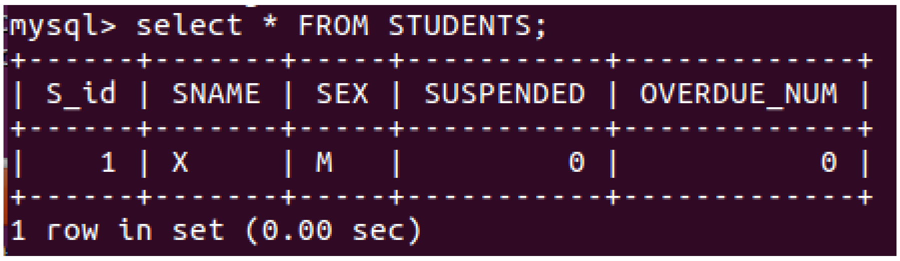
测试：

创建一个学号为１，姓名为Ｘ的男性学生：



```
Please enter your administrator id
1

What are you going to do?
1.Add book to library
2.Remove book from library
3.Add student account
4.Query books
5.Qurey borrow history
6.Qurey unreturned book
7.Check the deadline
8.Extend the deadline
9.Check overdue
10.Suspend student accounts
0.Back to showmenu
3

Please enter S_id, SNAME and SEX
1 X M
This student account has been successfully established!
```

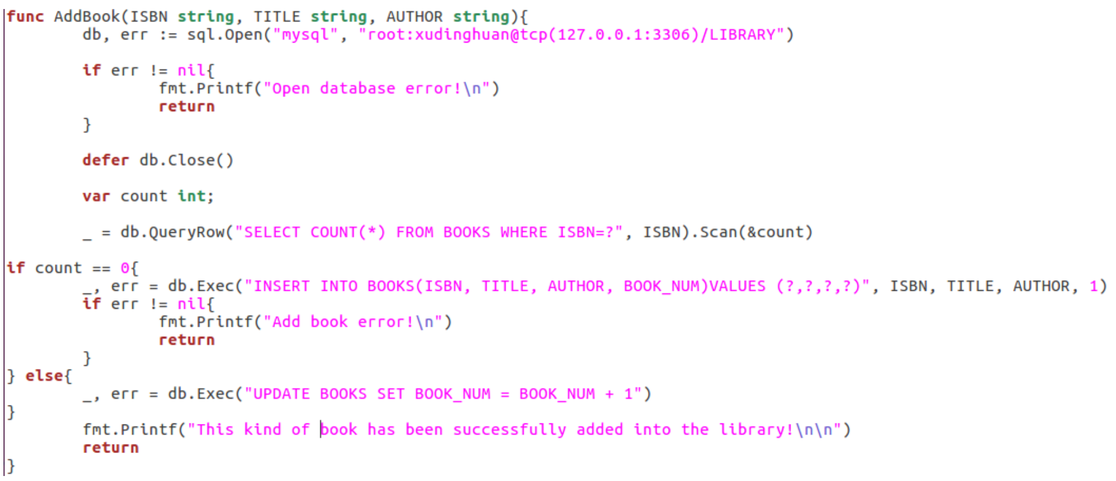
以１号管理员的身份执行３号功能输入信息后，在库中可以看到：



```
mysql> select * FROM STUDENTS;
+------+-------+-----+-----------+-------------+
| S_id | SNAME | SEX | SUSPENDED | OVERDUE_NUM |
+------+-------+-----+-----------+-------------+
|    1 | X     | M   |         0 |           0 |
+------+-------+-----+-----------+-------------+
1 row in set (0.00 sec)
```

的的确确创建了这么一个学生账户。

２.向馆中添加书的ＡｄｄＢｏｏｋ函数：



```
func AddBook(ISBN string, TITLE string, AUTHOR string){
        db, err := sql.Open("mysql", "root:xudinghuan@tcp(127.0.0.1:3306)/LIBRARY")

        if err != nil{
                fmt.Printf("Open database error!\n")
                return
        }

        defer db.Close()

        var count int;

        _ = db.QueryRow("SELECT COUNT(*) FROM BOOKS WHERE ISBN=?", ISBN).Scan(&count)

if count == 0{
        _, err = db.Exec("INSERT INTO BOOKS(ISBN, TITLE, AUTHOR, BOOK_NUM)VALUES (?,?,?,?)", ISBN, TITLE, AUTHOR, 1)
        if err != nil{
                fmt.Printf("Add book error!\n")
                return
        }
} else{
        _, err = db.Exec("UPDATE BOOKS SET BOOK_NUM = BOOK_NUM + 1")
}
        fmt.Printf("This kind of book has been successfully added into the library!\n\n")
        return
}
```

先用主键书号查询该类型的书是否已经在馆中，若是：直接书数自增；若不是：用插入语句插入。

测试：

添加６本书，属性顺序：书号、书名、作者

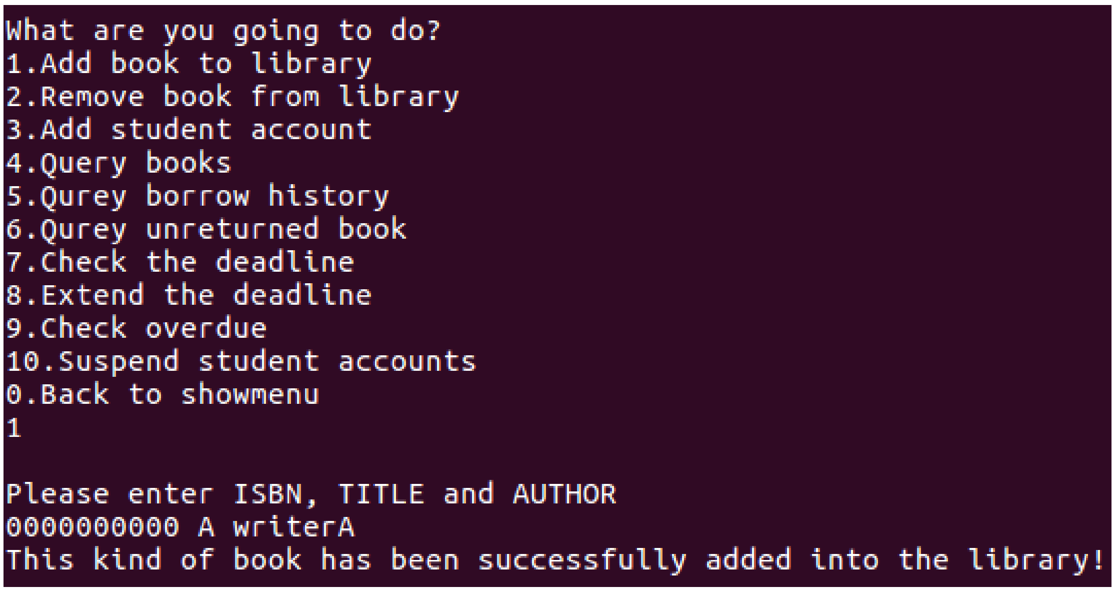
２本：００００００００００　Ａ　ｗｒｉｔｅｒＡ

其余４本：１１１１１１１１１１　Ｂ　ｗｒｉｔｅｒＢ

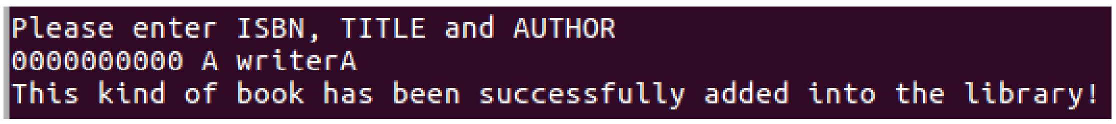
　　　　　２２２２２２２２２２　Ｃ　ｗｒｉｔｅｒＣ

　　　　　３３３３３３３３３３　Ｄ　ｗｒｉｔｅｒＤ

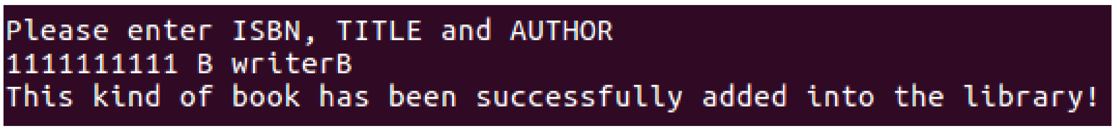
　　　　　４４４４４４４４４４　Ｅ　ｗｒｉｔｅｒＥ



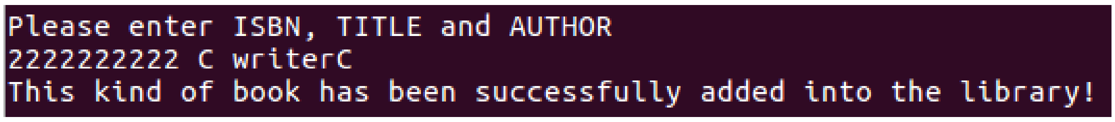
```
What are you going to do?
1.Add book to library
2.Remove book from library
3.Add student account
4.Query books
5.Qurey borrow history
6.Qurey unreturned book
7.Check the deadline
8.Extend the deadline
9.Check overdue
10.Suspend student accounts
0.Back to showmenu
1

Please enter ISBN, TITLE and AUTHOR
0000000000 A writerA
This kind of book has been successfully added into the library!
```



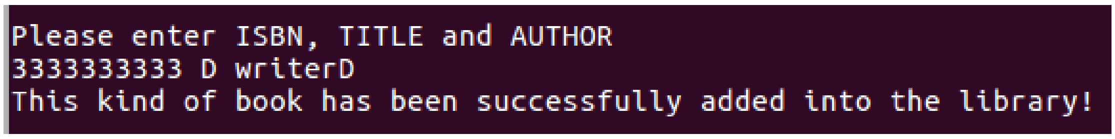
```
Please enter ISBN, TITLE and AUTHOR
0000000000 A writerA
This kind of book has been successfully added into the library!
```



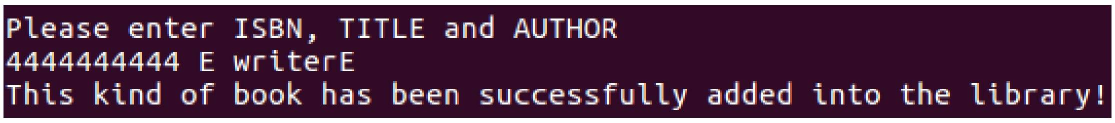
```
Please enter ISBN, TITLE and AUTHOR
1111111111 B writerB
This kind of book has been successfully added into the library!
```



```
Please enter ISBN, TITLE and AUTHOR
2222222222 C writerC
This kind of book has been successfully added into the library!
```

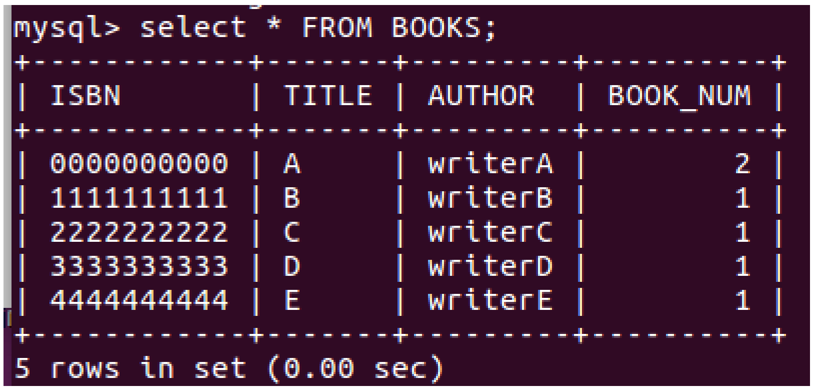


```
Please enter ISBN, TITLE and AUTHOR
3333333333 D writerD
This kind of book has been successfully added into the library!
```



```
Please enter ISBN, TITLE and AUTHOR
4444444444 E writerE
This kind of book has been successfully added into the library!
```

添加之后，可以看到库中：



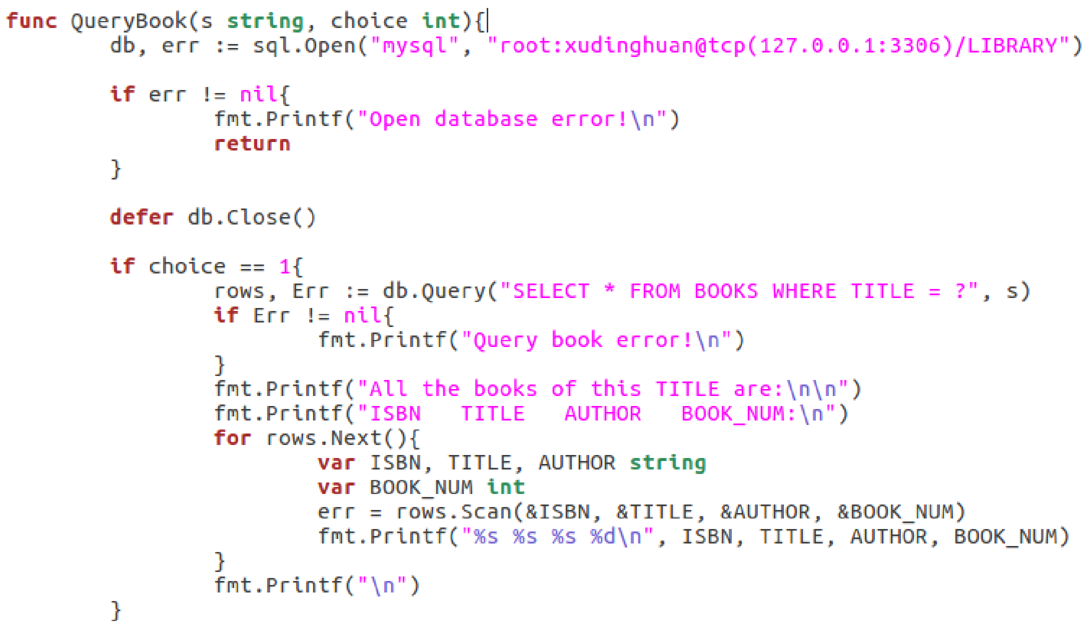
```
mysql> select * FROM BOOKS;
+------------+-------+---------+----------+
| ISBN       | TITLE | AUTHOR  | BOOK_NUM |
+------------+-------+---------+----------+
| 0000000000 | A     | writerA |        2 |
| 1111111111 | B     | writerB |        1 |
| 2222222222 | C     | writerC |        1 |
| 3333333333 | D     | writerD |        1 |
| 4444444444 | E     | writerE |        1 |
+------------+-------+---------+----------+
5 rows in set (0.00 sec)
```

3.查询书籍的ＱｕｅｒｙＢｏｏｋ函数：

在输出学生功能菜单的函数中对应查书功能的是这一段代码：

```
case 1:
        fmt.Printf("\nPlease enter TITLE or AUTHOR or ISBN:\n")
        fmt.Printf("1.Enter TITLE\n")
        fmt.Printf("2.Enter AUTHOR\n")
        fmt.Printf("3.Enter ISBN\n\n")
        var choicecase1 int
        fmt.Scanln(&choicecase1)
        var s string
        fmt.Printf("\nPlease enter the chosen information\n")
        fmt.Scanln(&s)
        QueryBook(s, choicecase1)
```

可以看到，在选择了这个功能之后，会先询问用户要用书的哪个属性进行查找，三个属性各对应一个数字，将用户选择输入的数字保存到变量中传递给功能函数：

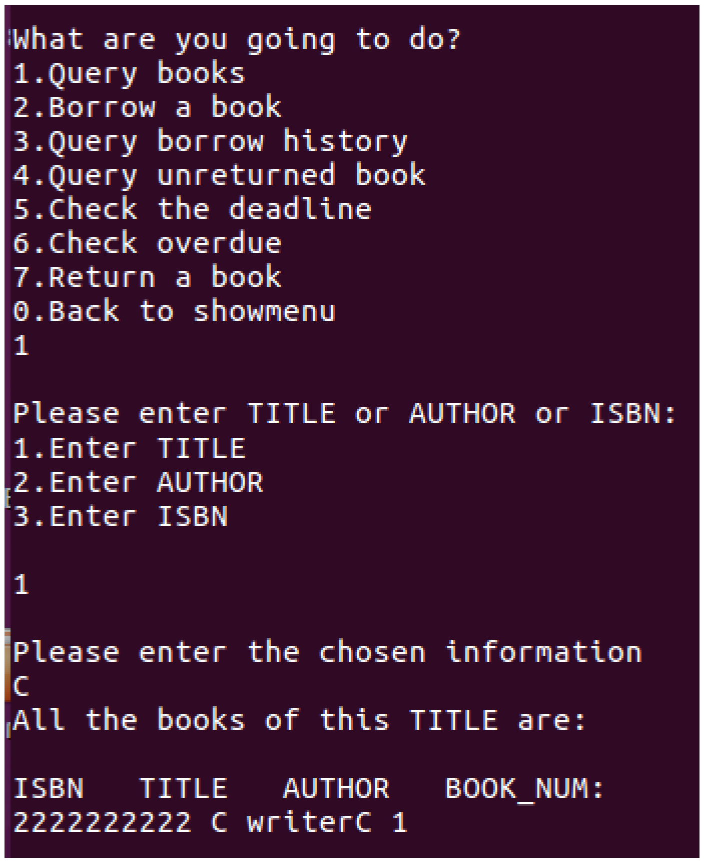


```
func QueryBook(s string, choice int){
	db, err := sql.Open("mysql", "root:xudinghuan@tcp(127.0.0.1:3306)/LIBRARY")

	if err != nil{
		fmt.Printf("Open database error!\n")
		return
	}

	defer db.Close()

	if choice == 1{
		rows, Err := db.Query("SELECT * FROM BOOKS WHERE TITLE = ?", s)
		if Err != nil{
			fmt.Printf("Query book error!\n")
		}
		fmt.Printf("All the books of this TITLE are:\n\n")
		fmt.Printf("ISBN    TITLE    AUTHOR    BOOK_NUM:\n")
		for rows.Next(){
			var ISBN, TITLE, AUTHOR string
			var BOOK_NUM int
			err = rows.Scan(&ISBN, &TITLE, &AUTHOR, &BOOK_NUM)
			fmt.Printf("%s %s %s %d\n", ISBN, TITLE, AUTHOR, BOOK_NUM)
		}
		fmt.Printf("\n")
	}
```

而在功能函数中，根据得到的用户选择数字ｃｈｏｉｃｅ决定执行哪段代码（由于其余两段代码高度相似，就不贴了）。执行一个对应属性的查询语句，后将结果遍历输出即可。

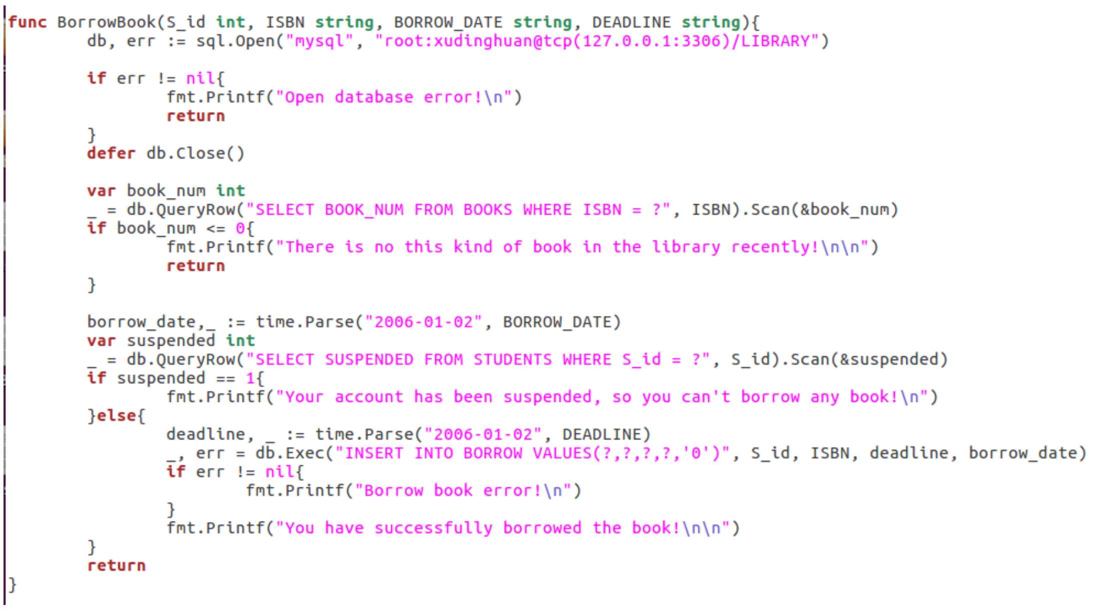
测试：

不妨就以学生的身份用书名查询：



```
What are you going to do?
1.Query books
2.Borrow a book
3.Query borrow history
4.Query unreturned book
5.Check the deadline
6.Check overdue
7.Return a book
0.Back to showmenu
1

Please enter TITLE or AUTHOR or ISBN:
1.Enter TITLE
2.Enter AUTHOR
3.Enter ISBN

1

Please enter the chosen information
C
All the books of this TITLE are:

ISBN    TITLE    AUTHOR    BOOK_NUM:
2222222222 C writerC 1
```

4.借书的ＢｏｒｒｏｗＢｏｏｋ函数：

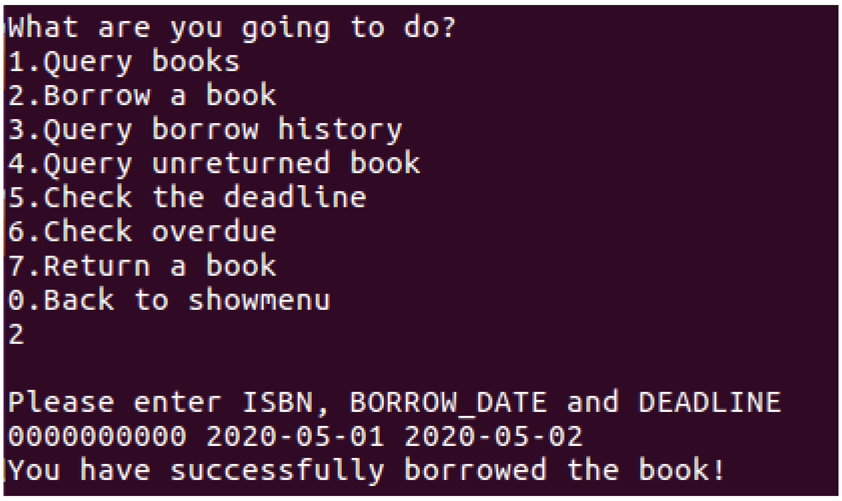


```go
func BorrowBook(S_id int, ISBN string, BORROW_DATE string, DEADLINE string){
	db, err := sql.Open("mysql", "root:xudinghuan@tcp(127.0.0.1:3306)/LIBRARY")

	if err != nil{
		fmt.Printf("Open database error!\n")
		return
	}
	defer db.Close()

	var book_num int
	_ = db.QueryRow("SELECT BOOK_NUM FROM BOOKS WHERE ISBN = ?", ISBN).Scan(&book_num)
	if book_num <= 0{
		fmt.Printf("There is no this kind of book in the library recently!\n\n")
		return
	}

	borrow_date,_ := time.Parse("2006-01-02", BORROW_DATE)
	var suspended int
	_ = db.QueryRow("SELECT SUSPENDED FROM STUDENTS WHERE S_id = ?", S_id).Scan(&suspended)
	if suspended == 1{
		fmt.Printf("Your account has been suspended, so you can't borrow any book!\n")
	}else{
		deadline, _ := time.Parse("2006-01-02", DEADLINE)
		_, err = db.Exec("INSERT INTO BORROW VALUES(?,?,?,?,'0')", S_id, ISBN, deadline, borrow_date)
		if err != nil{
			fmt.Printf("Borrow book error!\n")
		}
		fmt.Printf("You have successfully borrowed the book!\n\n")
	}
	return
}
```

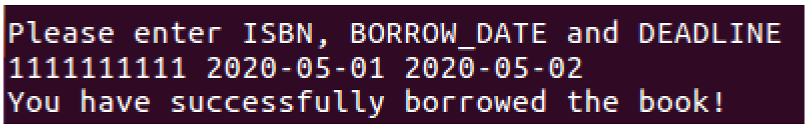
先查询馆中的相应书是否有足够的数量，再查询该学生账户是否被悬停，两者中任意一个不满足（没有书、被悬停），则不能借书。若可以借书，则在借阅记录中插入本次记录。

测试：

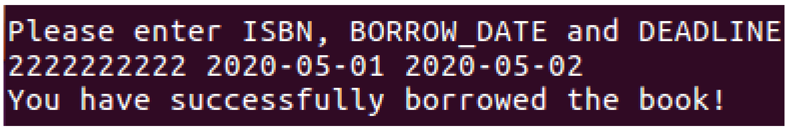
不妨已有的５类书各借一本，都于２０２０年５月１日借出，而ＤＤＬ均是该年５月２日（即均已超时）



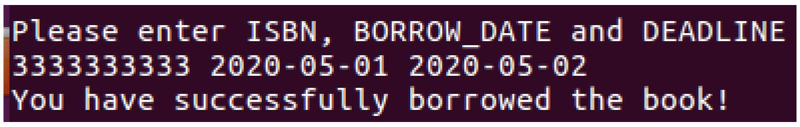
```
What are you going to do?
1.Query books
2.Borrow a book
3.Query borrow history
4.Query unreturned book
5.Check the deadline
6.Check overdue
7.Return a book
0.Back to showmenu
2

Please enter ISBN, BORROW_DATE and DEADLINE
0000000000 2020-05-01 2020-05-02
You have successfully borrowed the book!
```



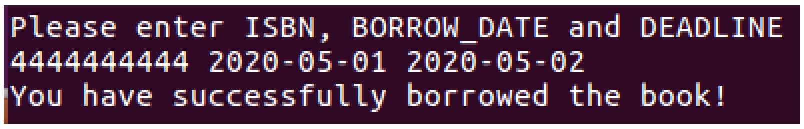
```
Please enter ISBN, BORROW_DATE and DEADLINE
1111111111 2020-05-01 2020-05-02
You have successfully borrowed the book!
```



```
Please enter ISBN, BORROW_DATE and DEADLINE
2222222222 2020-05-01 2020-05-02
You have successfully borrowed the book!
```

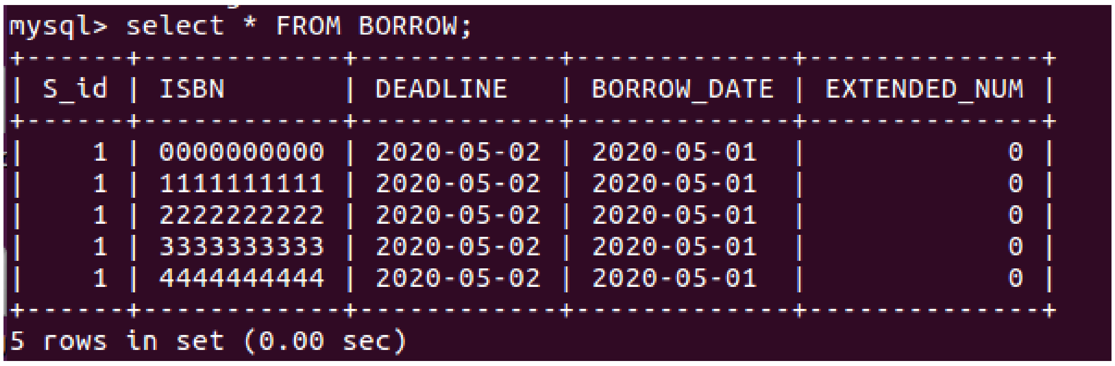


```
Please enter ISBN, BORROW_DATE and DEADLINE
3333333333 2020-05-01 2020-05-02
You have successfully borrowed the book!
```

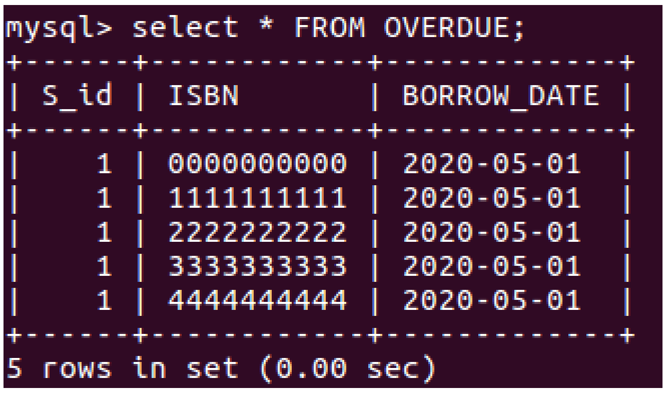


```
Please enter ISBN, BORROW_DATE and DEADLINE
4444444444 2020-05-01 2020-05-02
You have successfully borrowed the book!
```

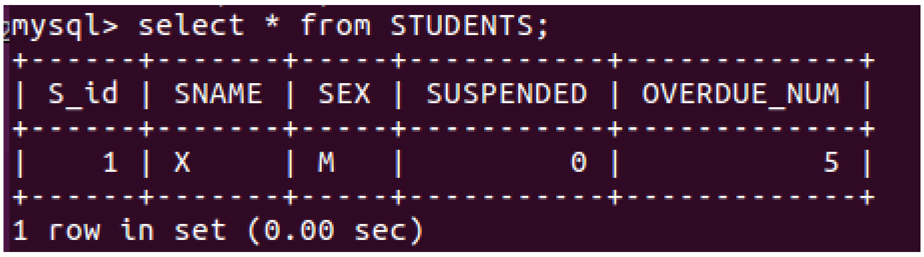
功能执行完毕后，查看库中状况：



```
mysql> select * FROM BORROW;
+------+------------+------------+-------------+--------------+
| S_id | ISBN       | DEADLINE   | BORROW_DATE | EXTENDED_NUM |
+------+------------+------------+-------------+--------------+
|    1 | 0000000000 | 2020-05-02 | 2020-05-01  |            0 |
|    1 | 1111111111 | 2020-05-02 | 2020-05-01  |            0 |
|    1 | 2222222222 | 2020-05-02 | 2020-05-01  |            0 |
|    1 | 3333333333 | 2020-05-02 | 2020-05-01  |            0 |
|    1 | 4444444444 | 2020-05-02 | 2020-05-01  |            0 |
+------+------------+------------+-------------+--------------+
5 rows in set (0.00 sec)
```

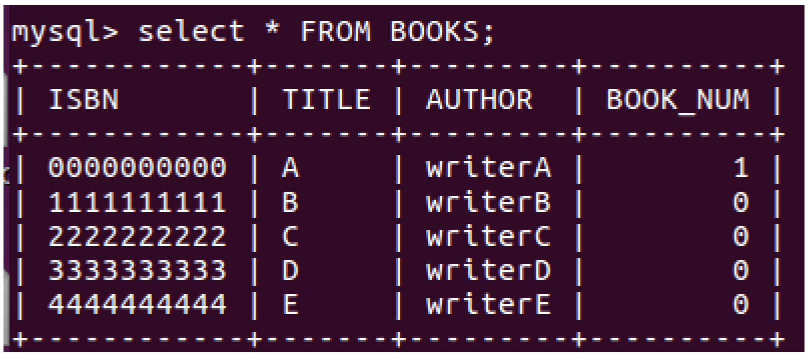


```
mysql> select * FROM OVERDUE;
+------+------------+-------------+
| S_id | ISBN       | BORROW_DATE |
+------+------------+-------------+
|    1 | 0000000000 | 2020-05-01  |
|    1 | 1111111111 | 2020-05-01  |
|    1 | 2222222222 | 2020-05-01  |
|    1 | 3333333333 | 2020-05-01  |
|    1 | 4444444444 | 2020-05-01  |
+------+------------+-------------+
5 rows in set (0.00 sec)
```



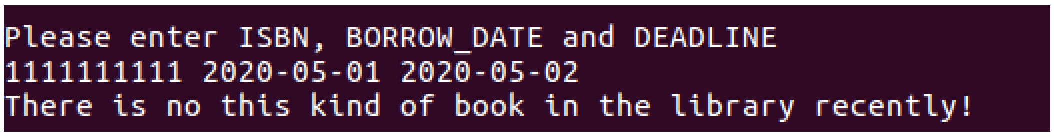
```
mysql> select * from STUDENTS;
+------+-------+-----+-----------+-------------+
| S_id | SNAME | SEX | SUSPENDED | OVERDUE_NUM |
+------+-------+-----+-----------+-------------+
|    1 | X     | M   |         0 |           5 |
+------+-------+-----+-----------+-------------+
1 row in set (0.00 sec)
```

借阅记录和超时处罚单都正确地显现了出来，学生的超时次数也被设置为 5（本人的理解是就算超时次数再多也需要管理员亲手悬停），再看表ＢＯＯＫＳ：



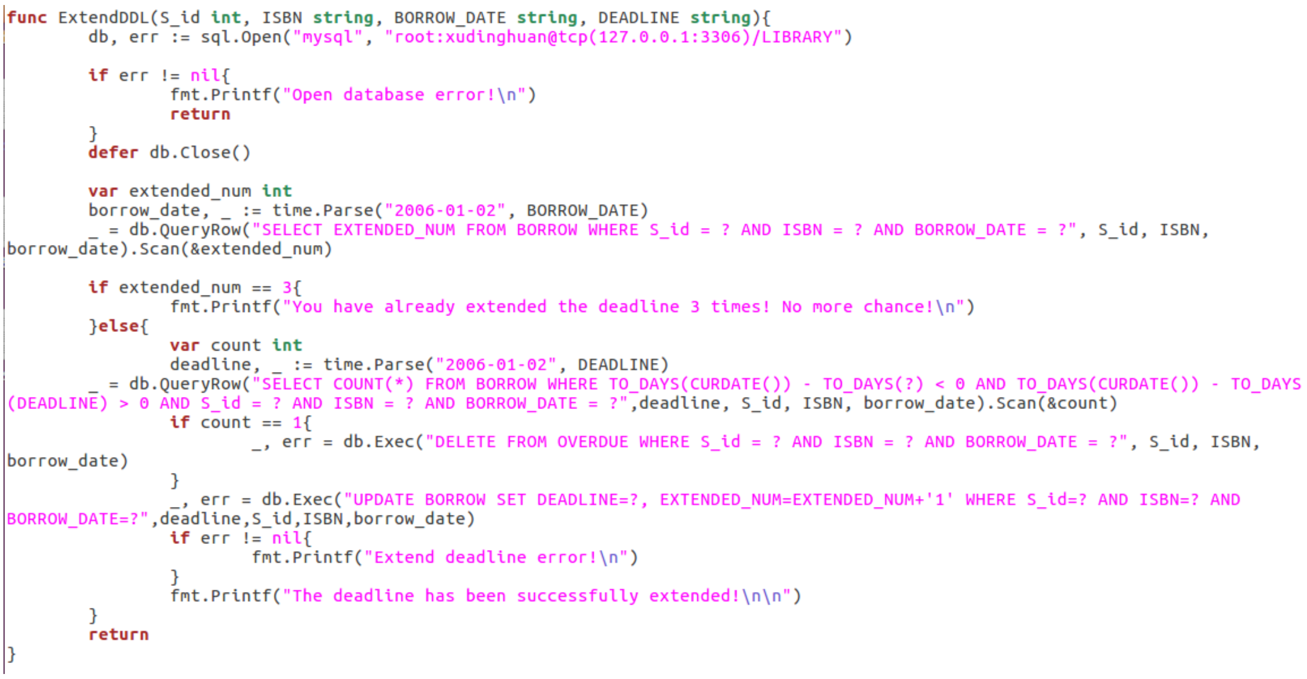
```
mysql> select * FROM BOOKS;
+------------+-------+---------+----------+
| ISBN       | TITLE | AUTHOR  | BOOK_NUM |
+------------+-------+---------+----------+
| 0000000000 | A     | writerA |        1 |
| 1111111111 | B     | writerB |        0 |
| 2222222222 | C     | writerC |        0 |
| 3333333333 | D     | writerD |        0 |
| 4444444444 | E     | writerE |        0 |
+------------+-------+---------+----------+
```

每本被借走的书的余量也都减了１，此时去借没有余量的书，则会发出借书失败信息：



```
Please enter ISBN, BORROW_DATE and DEADLINE
1111111111 2020-05-01 2020-05-02
There is no this kind of book in the library recently!
```

5.延长ｄｄｌ的ＥｘｔｅｎｄＤＤＬ函数：

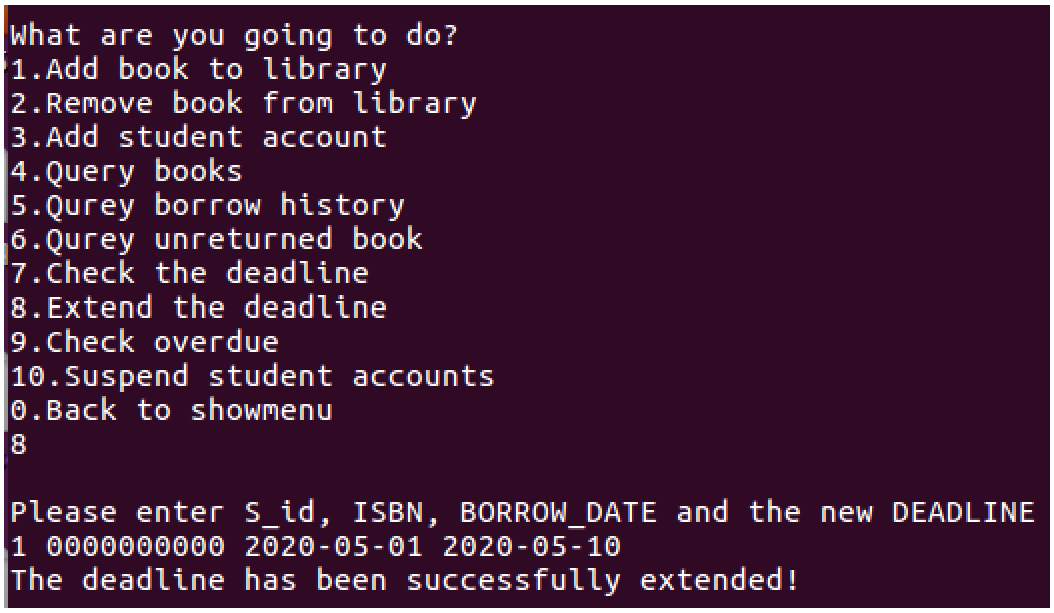


```
func ExtendDDL(S_id int, ISBN string, BORROW_DATE string, DEADLINE string){
	db, err := sql.Open("mysql", "root:xudinghuan@tcp(127.0.0.1:3306)/LIBRARY")

	if err != nil{
		fmt.Printf("Open database error!\n")
		return
	}
	defer db.Close()

	var extended_num int
	borrow_date, _ := time.Parse("2006-01-02", BORROW_DATE)
	_ = db.QueryRow("SELECT EXTENDED_NUM FROM BORROW WHERE S_id = ? AND ISBN = ? AND BORROW_DATE = ?", S_id, ISBN,
borrow_date).Scan(&extended_num)

	if extended_num == 3{
		fmt.Printf("You have already extended the deadline 3 times! No more chance!\n")
	}else{
		var count int
		deadline, _ := time.Parse("2006-01-02", DEADLINE)
	_ = db.QueryRow("SELECT COUNT(*) FROM BORROW WHERE TO_DAYS(CURDATE()) - TO_DAYS(?) < 0 AND TO_DAYS(CURDATE()) - TO_DAYS
(DEADLINE) > 0 AND S_id = ? AND ISBN = ? AND BORROW_DATE = ?",deadline, S_id, ISBN, borrow_date).Scan(&count)
		if count == 1{
			_, err = db.Exec("DELETE FROM OVERDUE WHERE S_id = ? AND ISBN = ? AND BORROW_DATE = ?", S_id, ISBN,
borrow_date)
		}
		_, err = db.Exec("UPDATE BORROW SET DEADLINE=?, EXTENDED_NUM=EXTENDED_NUM+'1' WHERE S_id=? AND ISBN=? AND
BORROW_DATE=?",deadline,S_id,ISBN,borrow_date)
		if err != nil{
			fmt.Printf("Extend deadline error!\n")
		}
		fmt.Printf("The deadline has been successfully extended!\n\n")
	}
	return
}
```

在该函数中，首先使用查询语句查询借阅记录中ｄｄｌ的延长次数，如果已经延长了３次，则拒绝此次延长请求。由于存在一种特殊情形，即学生在超时未归还从而生成了罚单时延长了ｄｄｌ，此时合理的做法应该是撤销罚单（学生超时次数会自动减１），于是再次执行一个查询语句，通过查看当前日期＞老ｄｄｌ与当前日期＜新ｄｄｌ（可判断是否是上述情形）的元组是否存在来决定后续操作。若存在，则需要从超时处罚单中撤销相应的罚单，并修改借阅记录中的ｄｄｌ；若不存在，只修改ｄｄｌ即可。

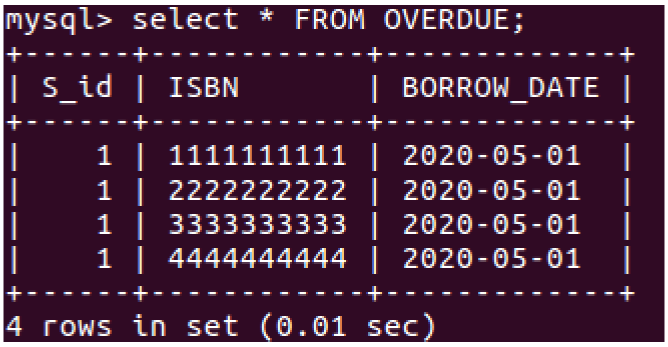
测试：

不妨测试一下上述特殊情形，即把００００００００００这本书（已超时）的ｄｄｌ改为２０２０－０５－１０（由于报告撰写时间是５月３日，显然此时不超时）：



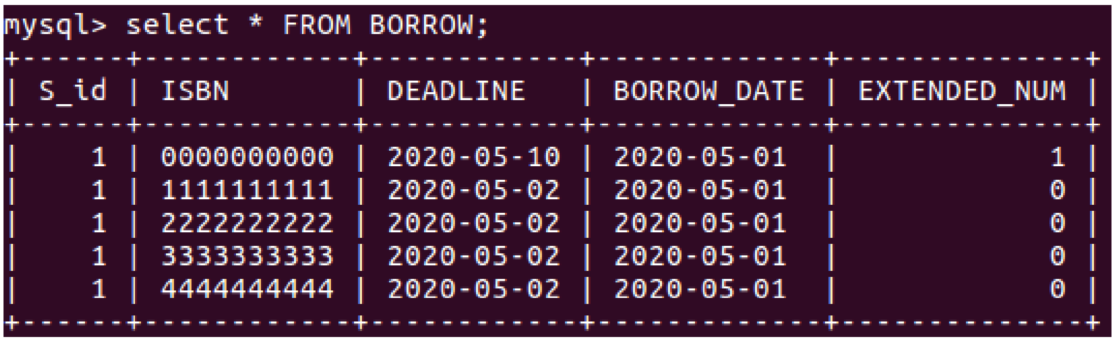
```
What are you going to do?
1.Add book to library
2.Remove book from library
3.Add student account
4.Query books
5.Qurey borrow history
6.Qurey unreturned book
7.Check the deadline
8.Extend the deadline
9.Check overdue
10.Suspend student accounts
0.Back to showmenu
8

Please enter S_id, ISBN, BORROW_DATE and the new DEADLINE
1 0000000000 2020-05-01 2020-05-10
The deadline has been successfully extended!
```

查看此时库中状况：



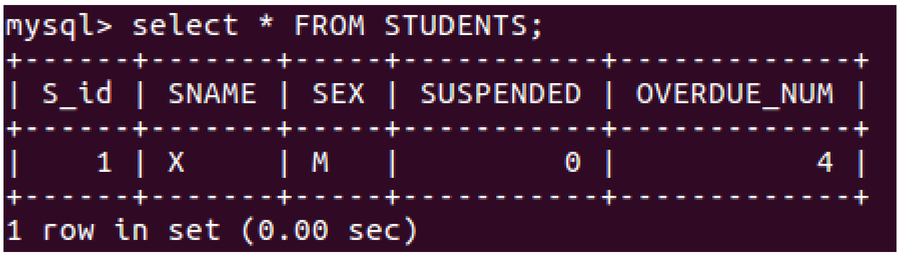
```
mysql> select * FROM OVERDUE;
+------+------------+-------------+
| S_id | ISBN       | BORROW_DATE |
+------+------------+-------------+
|    1 | 1111111111 | 2020-05-01  |
|    1 | 2222222222 | 2020-05-01  |
|    1 | 3333333333 | 2020-05-01  |
|    1 | 4444444444 | 2020-05-01  |
+------+------------+-------------+
4 rows in set (0.01 sec)
```

罚单着实撤销了对应的那一条

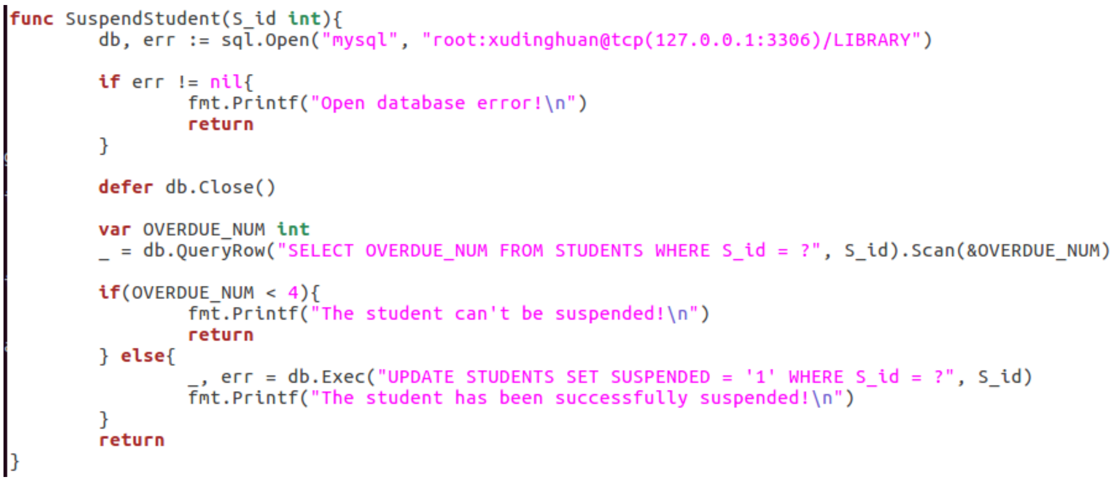


```
mysql> select * FROM BORROW;
+------+------------+------------+-------------+--------------+
| S_id | ISBN       | DEADLINE   | BORROW_DATE | EXTENDED_NUM |
+------+------------+------------+-------------+--------------+
|    1 | 0000000000 | 2020-05-10 | 2020-05-01  |            1 |
|    1 | 1111111111 | 2020-05-02 | 2020-05-01  |            0 |
|    1 | 2222222222 | 2020-05-02 | 2020-05-01  |            0 |
|    1 | 3333333333 | 2020-05-02 | 2020-05-01  |            0 |
|    1 | 4444444444 | 2020-05-02 | 2020-05-01  |            0 |
+------+------------+------------+-------------+--------------+
```

借阅记录中第一本书的ｄｄｌ得到了延长，延长次数也增加了１。

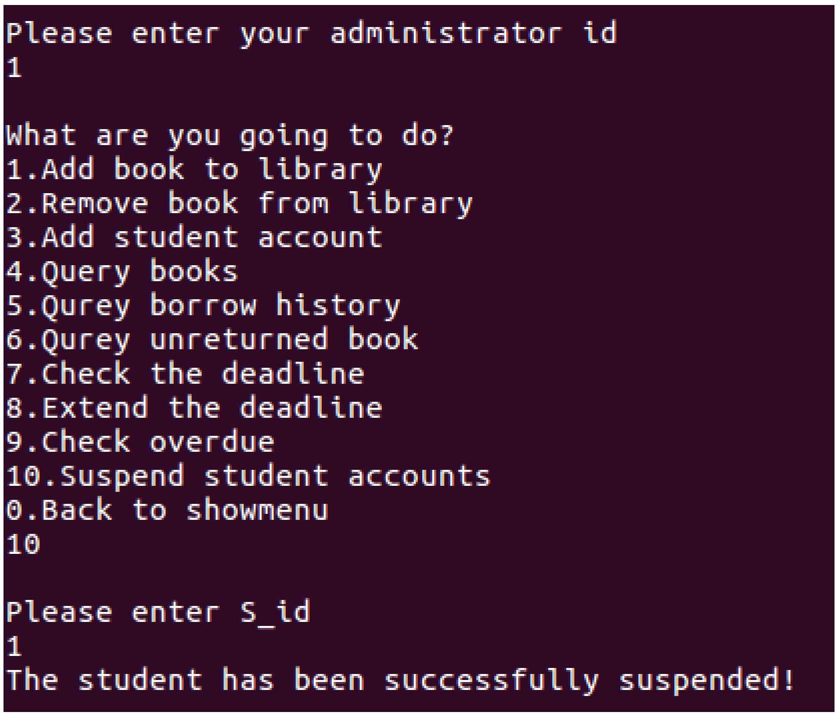

```
mysql> select * FROM STUDENTS;
+------+-------+-----+-----------+-------------+
| S_id | SNAME | SEX | SUSPENDED | OVERDUE_NUM |
+------+-------+-----+-----------+-------------+
|    1 | X     | M   |         0 |           4 |
+------+-------+-----+-----------+-------------+
1 row in set (0.00 sec)
```
学生的超时次数也减少了１。

６.悬停学生的ＳｕｓｐｅｎｄＳｔｕｄｅｎｔ函数：

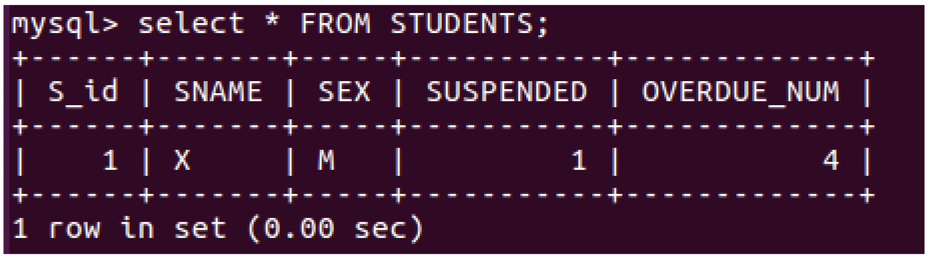

```go
func SuspendStudent(S_id int){
	db, err := sql.Open("mysql", "root:xudinghuan@tcp(127.0.0.1:3306)/LIBRARY")

	if err != nil{
		fmt.Printf("Open database error!\n")
		return
	}

	defer db.Close()

	var OVERDUE_NUM int
	_ = db.QueryRow("SELECT OVERDUE_NUM FROM STUDENTS WHERE S_id = ?", S_id).Scan(&OVERDUE_NUM)

	if(OVERDUE_NUM < 4){
		fmt.Printf("The student can't be suspended!\n")
		return
	} else{
		_, err = db.Exec("UPDATE STUDENTS SET SUSPENDED = '1' WHERE S_id = ?", S_id)
		fmt.Printf("The student has been successfully suspended!\n")
	}
	return
}
```
先使用查询语句查看学生的超时次数，若超时次数＜＝３，则不允悬停；若不然，则允许悬停，将相应学生的是否悬停这一属性改为１即可。

测试：

此时的１号学生应该是可以悬停的，将之悬停：

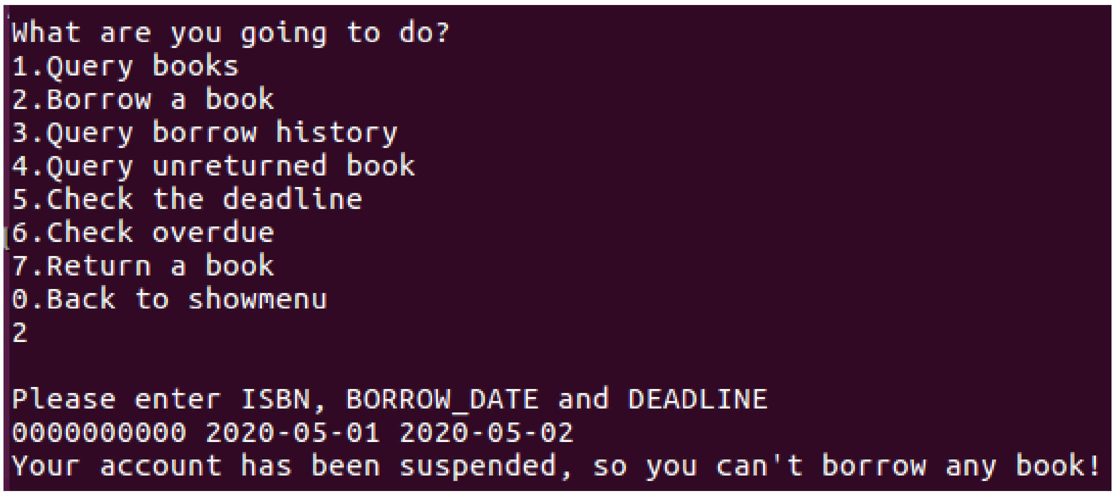

```
Please enter your administrator id
1

What are you going to do?
1.Add book to library
2.Remove book from library
3.Add student account
4.Query books
5.Qurey borrow history
6.Qurey unreturned book
7.Check the deadline
8.Extend the deadline
9.Check overdue
10.Suspend student accounts
0.Back to showmenu
10

Please enter S_id
1
The student has been successfully suspended!
```

执行功能后，库中状态：



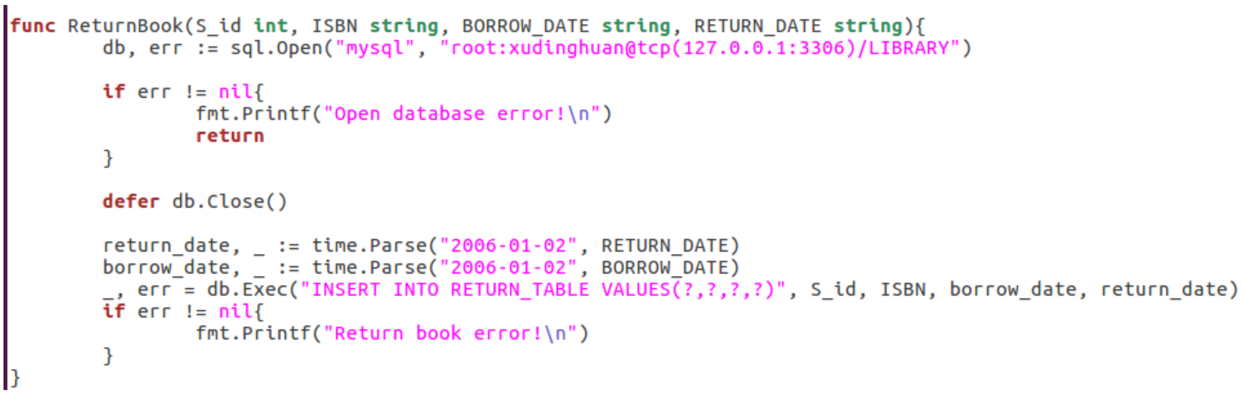
```
mysql> select * FROM STUDENTS;
+------+-------+-----+-----------+-------------+
| S_id | SNAME | SEX | SUSPENDED | OVERDUE_NUM |
+------+-------+-----+-----------+-------------+
|    1 | X     | M   |         1 |           4 |
+------+-------+-----+-----------+-------------+
1 row in set (0.00 sec)
```

学生被成功悬停，此时若该学生前往借书，则会被拒绝：



```
What are you going to do?
1.Query books
2.Borrow a book
3.Query borrow history
4.Query unreturned book
5.Check the deadline
6.Check overdue
7.Return a book
0.Back to showmenu
2

Please enter ISBN, BORROW_DATE and DEADLINE
0000000000 2020-05-01 2020-05-02
Your account has been suspended, so you can't borrow any book!
```

7.还书的ＲｅｔｕｒｎＢｏｏｋ函数：

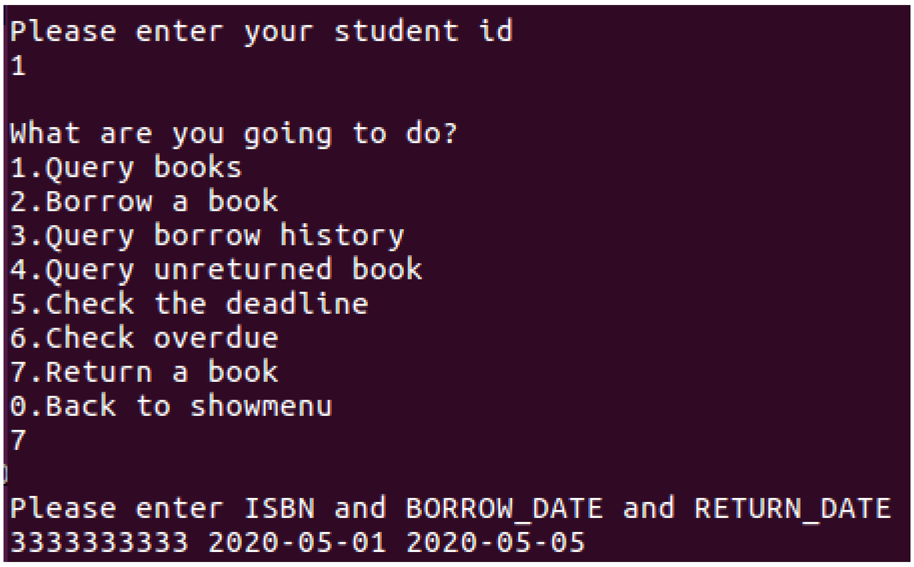


```go
func ReturnBook(S_id int, ISBN string, BORROW_DATE string, RETURN_DATE string){
        db, err := sql.Open("mysql", "root:xudinghuan@tcp(127.0.0.1:3306)/LIBRARY")

        if err != nil{
                fmt.Printf("Open database error!\n")
                return
        }

        defer db.Close()

        return_date, _ := time.Parse("2006-01-02", RETURN_DATE)
        borrow_date, _ := time.Parse("2006-01-02", BORROW_DATE)
        _, err = db.Exec("INSERT INTO RETURN_TABLE VALUES(?,?,?,?)", S_id, ISBN, borrow_date, return_date)
        if err != nil{
                fmt.Printf("Return book error!\n")
        }
}
```

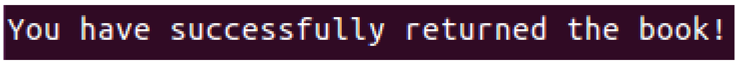
直接将学生输入的归还书籍信息插入归还记录中即可。

测试：

不妨于２０２０年５月５日归还３３３３３３３３３３这本超时的书：

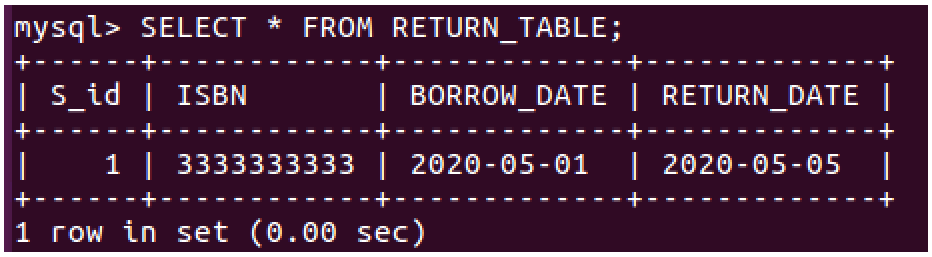


```
Please enter your student id
1

What are you going to do?
1.Query books
2.Borrow a book
3.Query borrow history
4.Query unreturned book
5.Check the deadline
6.Check overdue
7.Return a book
0.Back to showmenu
7

Please enter ISBN and BORROW_DATE and RETURN_DATE
3333333333 2020-05-01 2020-05-05
```



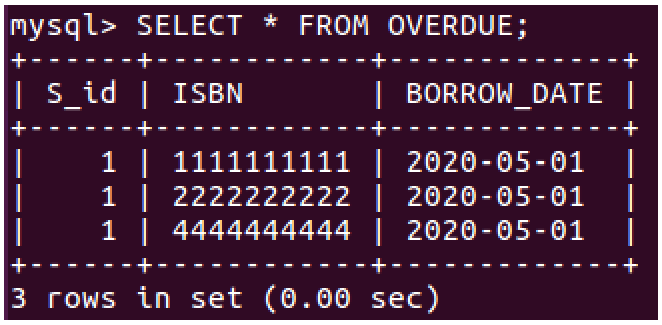
```
You have successfully returned the book!
```

此时，库中状态：



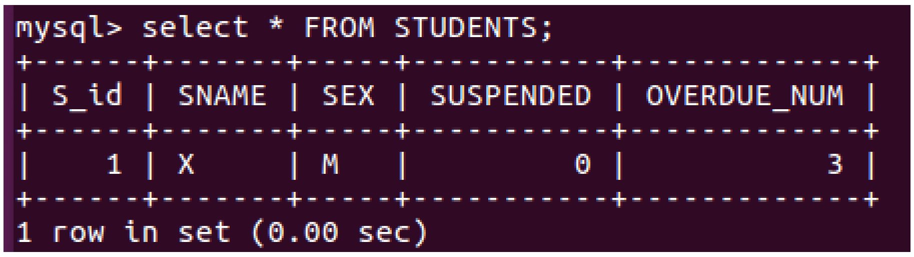
```
mysql> SELECT * FROM RETURN_TABLE;
+------+------------+-------------+-------------+
| S_id | ISBN       | BORROW_DATE | RETURN_DATE |
+------+------------+-------------+-------------+
|    1 | 3333333333 | 2020-05-01  | 2020-05-05  |
+------+------------+-------------+-------------+
1 row in set (0.00 sec)
```

可以看到，归还记录中增添了应有的一行。



```
mysql> SELECT * FROM OVERDUE;
+------+------------+-------------+
| S_id | ISBN       | BORROW_DATE |
+------+------------+-------------+
|    1 | 1111111111 | 2020-05-01  |
|    1 | 2222222222 | 2020-05-01  |
|    1 | 4444444444 | 2020-05-01  |
+------+------------+-------------+
3 rows in set (0.00 sec)
```

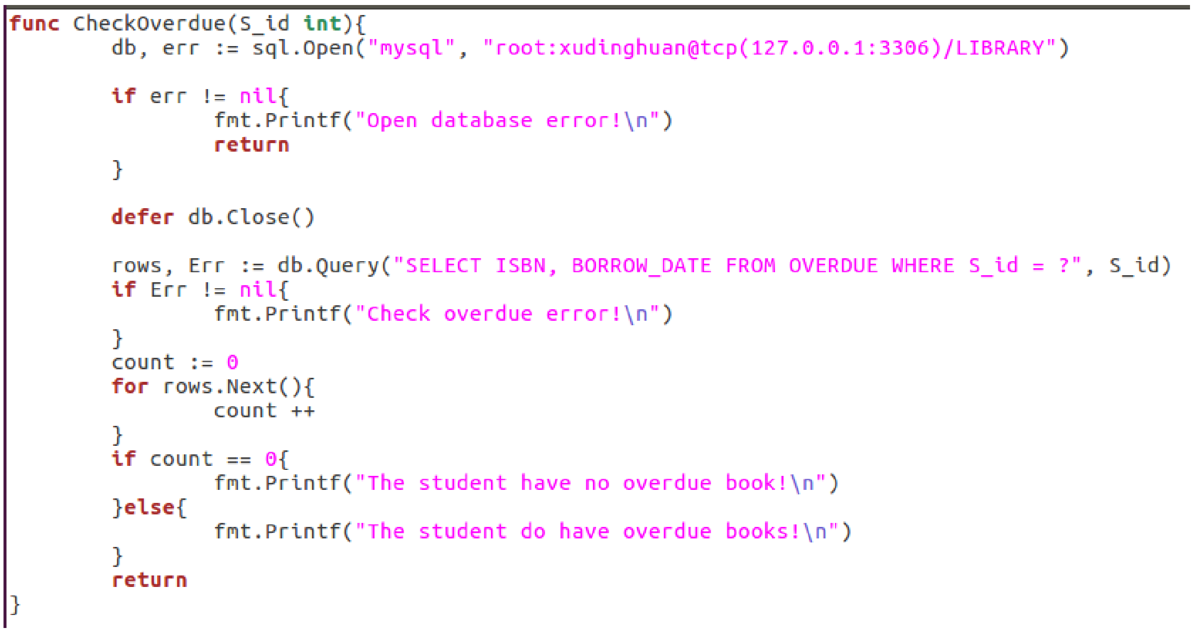
归还书本的罚单也撤销了。



```
mysql> select * FROM STUDENTS;
+------+-------+-----+-----------+-------------+
| S_id | SNAME | SEX | SUSPENDED | OVERDUE_NUM |
+------+-------+-----+-----------+-------------+
|    1 | X     | M   |         0 |           3 |
+------+-------+-----+-----------+-------------+
1 row in set (0.00 sec)
```

学生的超时次数自减，悬停状态被解除。

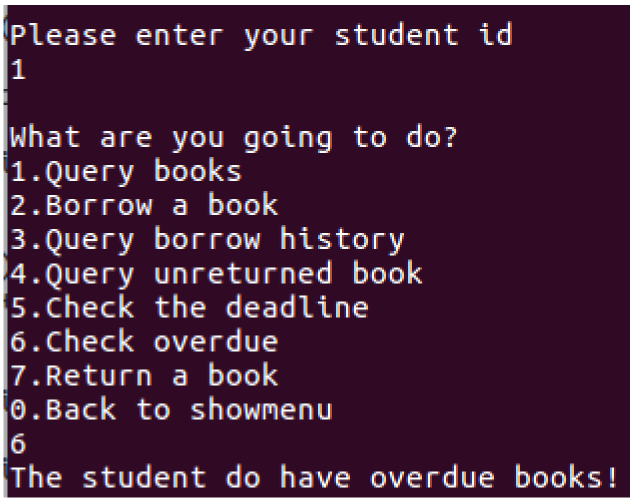
8.查看是否有超时书的CheckOverdue函数：



```
func CheckOverdue(S_id int){
	db, err := sql.Open("mysql", "root:xudinghuan@tcp(127.0.0.1:3306)/LIBRARY")

	if err != nil{
		fmt.Printf("Open database error!\n")
		return
	}

	defer db.Close()

	rows, Err := db.Query("SELECT ISBN, BORROW_DATE FROM OVERDUE WHERE S_id = ?", S_id)
	if Err != nil{
		fmt.Printf("Check overdue error!\n")
	}
	count := 0
	for rows.Next(){
		count ++
	}
	if count == 0{
		fmt.Printf("The student have no overdue book!\n")
	}else{
		fmt.Printf("The student do have overdue books!\n")
	}
	return
}
```

再超时惩罚中使用查询语句查找该学生的学号对应的元组，若查询结果为空，输出该生没有超时的书；若非空，输出该生有超时的书。

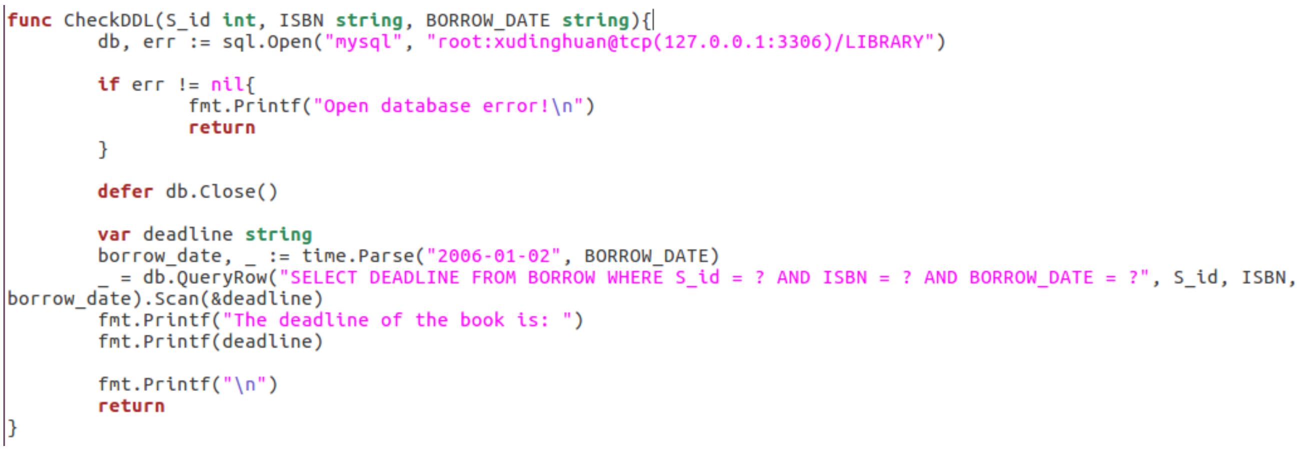
测试：



```
Please enter your student id
1

What are you going to do?
1.Query books
2.Borrow a book
3.Query borrow history
4.Query unreturned book
5.Check the deadline
6.Check overdue
7.Return a book
0.Back to showmenu
6
The student do have overdue books!
```

可以看到输出了该学生有超时的书。

９．查询ＤＤＬ的ＣｈｅｃｋＤＤＬ函数：

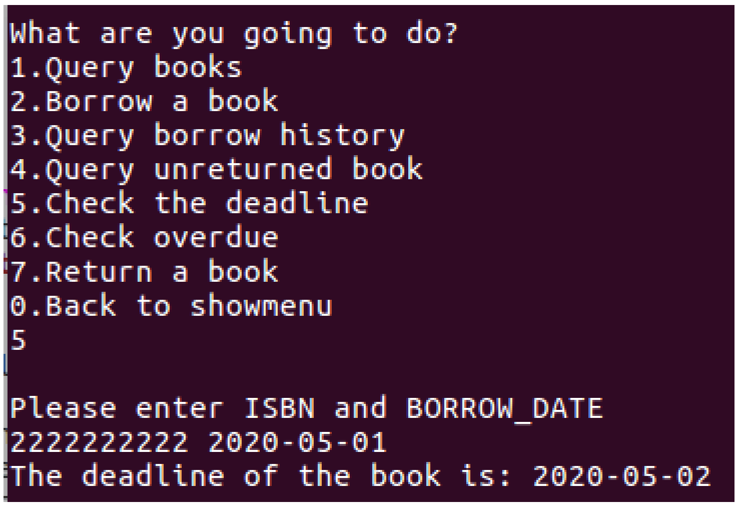


```go
func CheckDDL(S_id int, ISBN string, BORROW_DATE string){
	db, err := sql.Open("mysql", "root:xudinghuan@tcp(127.0.0.1:3306)/LIBRARY")

	if err != nil{
		fmt.Printf("Open database error!\n")
		return
	}

	defer db.Close()

	var deadline string
	borrow_date, _ := time.Parse("2006-01-02", BORROW_DATE)
	_ = db.QueryRow("SELECT DEADLINE FROM BORROW WHERE S_id = ? AND ISBN = ? AND BORROW_DATE = ?", S_id, ISBN, 
borrow_date).Scan(&deadline)
	fmt.Printf("The deadline of the book is: ")
	fmt.Printf(deadline)

	fmt.Printf("\n")
	return
}
```

直接用输入的主键查询并输出结果即可。

测试：

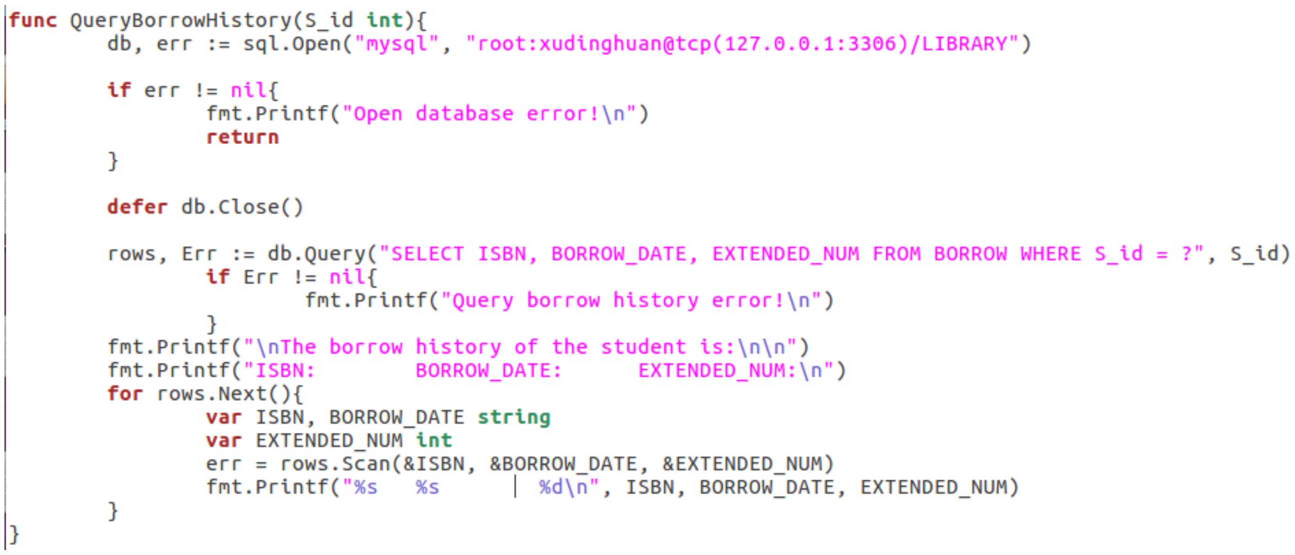
不妨查询５月１日借出给１同学的２２２２２２２２２２这本书的ＤＤＬ（应该是５月２日）



```
What are you going to do?
1.Query books
2.Borrow a book
3.Query borrow history
4.Query unreturned book
5.Check the deadline
6.Check overdue
7.Return a book
0.Back to showmenu
5

Please enter ISBN and BORROW_DATE
2222222222 2020-05-01
The deadline of the book is: 2020-05-02
```

得到了正确的结果。

１０.查询学生借阅记录的ＱｕｅｒｙＢｏｒｒｏｗＨｉｓｔｏｒｙ函数：

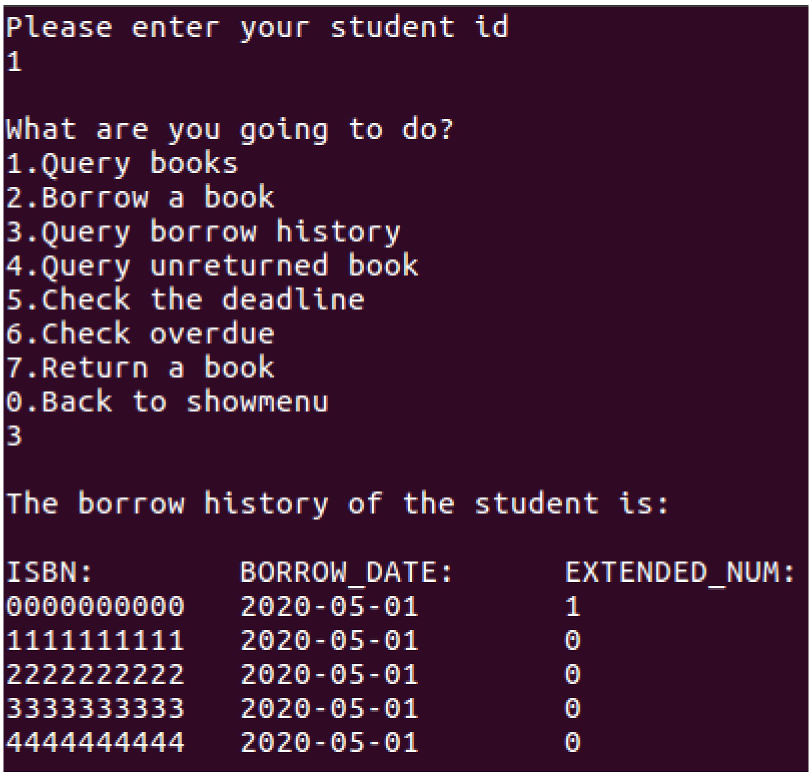


```go
func QueryBorrowHistory(S_id int){
	db, err := sql.Open("mysql", "root:xudinghuan@tcp(127.0.0.1:3306)/LIBRARY")

	if err != nil{
		fmt.Printf("Open database error!\n")
		return
	}

	defer db.Close()

	rows, Err := db.Query("SELECT ISBN, BORROW_DATE, EXTENDED_NUM FROM BORROW WHERE S_id = ?", S_id)
		if Err != nil{
			fmt.Printf("Query borrow history error!\n")
		}
	fmt.Printf("\nThe borrow history of the student is:\n\n")
	fmt.Printf("ISBN:        BORROW_DATE:      EXTENDED_NUM:\n")
	for rows.Next(){
		var ISBN, BORROW_DATE string
		var EXTENDED_NUM int
		err = rows.Scan(&ISBN, &BORROW_DATE, &EXTENDED_NUM)
		fmt.Printf("%s   %s      | %d\n", ISBN, BORROW_DATE, EXTENDED_NUM)
	}
}
```

用输入的学生学号在借阅记录中进行查询，再将查询结果遍历输出。

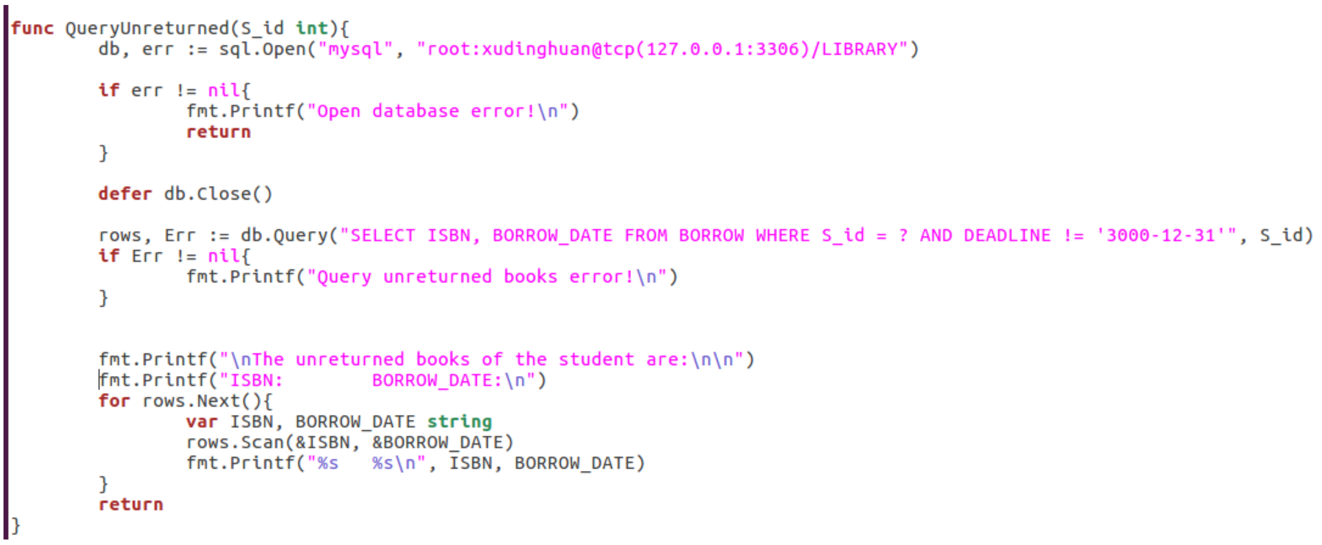
测试：

查看学生 1 的借阅记录：



```
Please enter your student id
1

What are you going to do?
1.Query books
2.Borrow a book
3.Query borrow history
4.Query unreturned book
5.Check the deadline
6.Check overdue
7.Return a book
0.Back to showmenu
3

The borrow history of the student is:

ISBN:        BORROW_DATE:      EXTENDED_NUM:
0000000000   2020-05-01        1
1111111111   2020-05-01        0
2222222222   2020-05-01        0
3333333333   2020-05-01        0
4444444444   2020-05-01        0
```

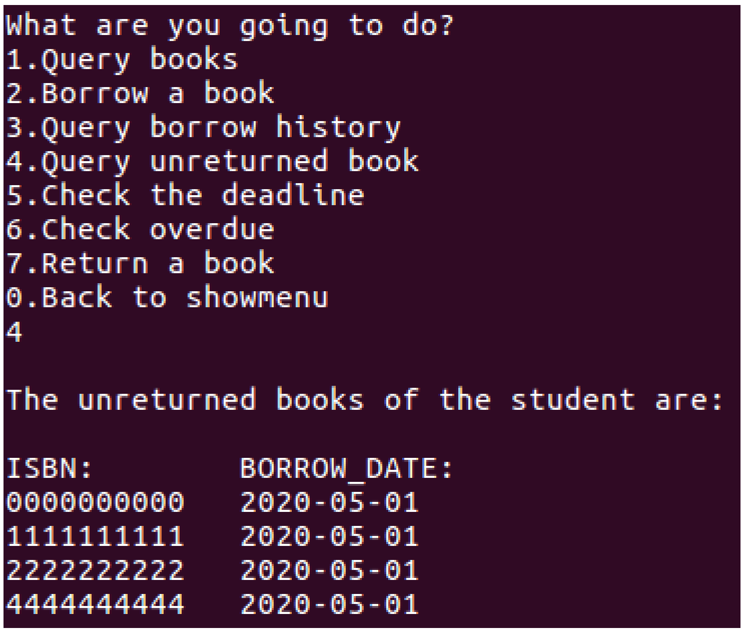
１１.查询尚未归还的书的ＱｕｅｒｙＵｎｒｅｔｕｒｎｅｄ函数：



```go
func QueryUnreturned(S_id int){
	db, err := sql.Open("mysql", "root:xudinghuan@tcp(127.0.0.1:3306)/LIBRARY")

	if err != nil{
		fmt.Printf("Open database error!\n")
		return
	}

	defer db.Close()

	rows, Err := db.Query("SELECT ISBN, BORROW_DATE FROM BORROW WHERE S_id = ? AND DEADLINE != '3000-12-31'", S_id)
	if Err != nil{
		fmt.Printf("Query unreturned books error!\n")
	}


	fmt.Printf("\nThe unreturned books of the student are:\n\n")
	fmt.Printf("ISBN:        BORROW_DATE:\n")
	for rows.Next(){
		var ISBN, BORROW_DATE string
		rows.Scan(&ISBN, &BORROW_DATE)
		fmt.Printf("%s   %s\n", ISBN, BORROW_DATE)
	}
	return
}
```

因为已经归还的书的ＤＤＬ会被修改为一个遥远的定值，因此在借阅记录中查询该生学号对应的元组中ＤＤＬ不为那个定值的元组，并将结果遍历输出即可。

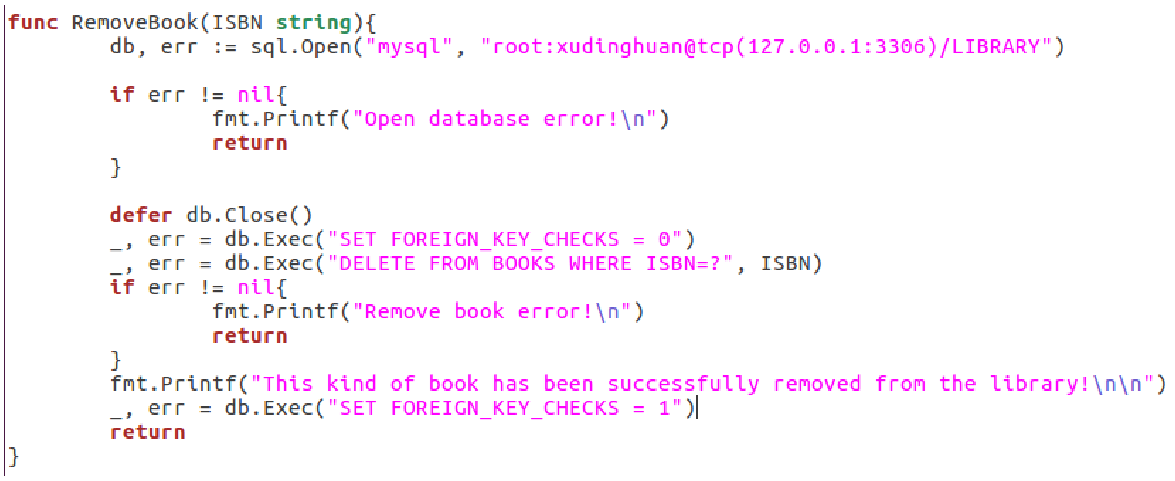
测试：



```
What are you going to do?
1.Query books
2.Borrow a book
3.Query borrow history
4.Query unreturned book
5.Check the deadline
6.Check overdue
7.Return a book
0.Back to showmenu
4

The unreturned books of the student are:

ISBN:        BORROW_DATE:
0000000000   2020-05-01
1111111111   2020-05-01
2222222222   2020-05-01
4444444444   2020-05-01
```

除了已经归还的３３３３３３３３３３号书，其余所有尚未归还的书都被输出了出来。

１２.从馆中将书除名的ＲｅｍｏｖｅＢｏｏｋ函数：

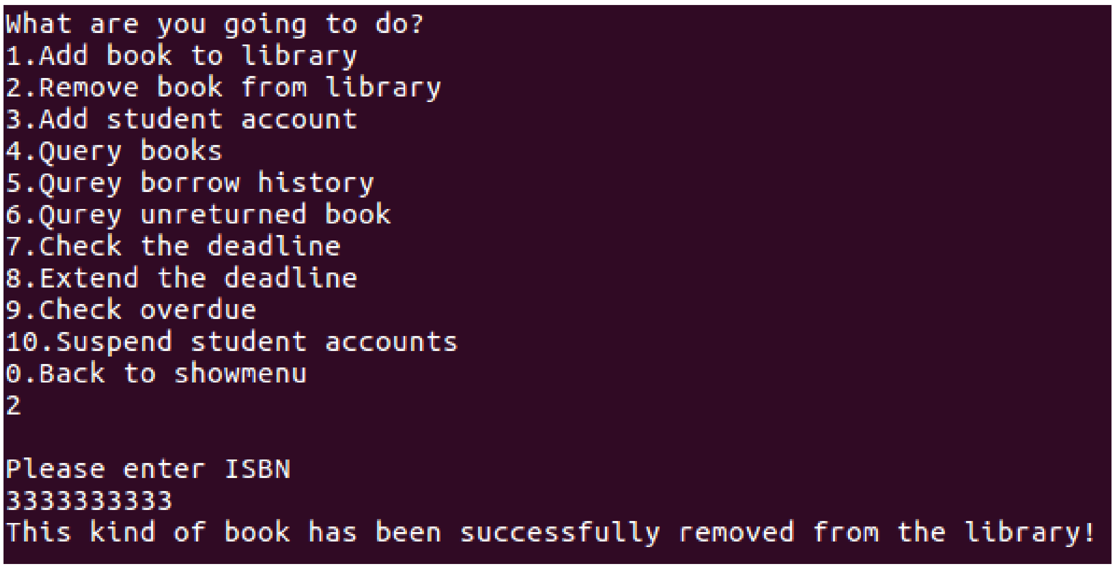


```go
func RemoveBook(ISBN string){
	db, err := sql.Open("mysql", "root:xudinghuan@tcp(127.0.0.1:3306)/LIBRARY")

	if err != nil{
		fmt.Printf("Open database error!\n")
		return
	}

	defer db.Close()
	_, err = db.Exec("SET FOREIGN_KEY_CHECKS = 0")
	_, err = db.Exec("DELETE FROM BOOKS WHERE ISBN=?", ISBN)
	if err != nil{
		fmt.Printf("Remove book error!\n")
		return
	}
	fmt.Printf("This kind of book has been successfully removed from the library!\n\n")
	_, err = db.Exec("SET FOREIGN_KEY_CHECKS = 1")
	return
}
```

由于书受到借阅记录、归还记录等的外键约束，故先将外键约束关闭，用删除语句从书中删除相应书号的元组，最后恢复外键约束。

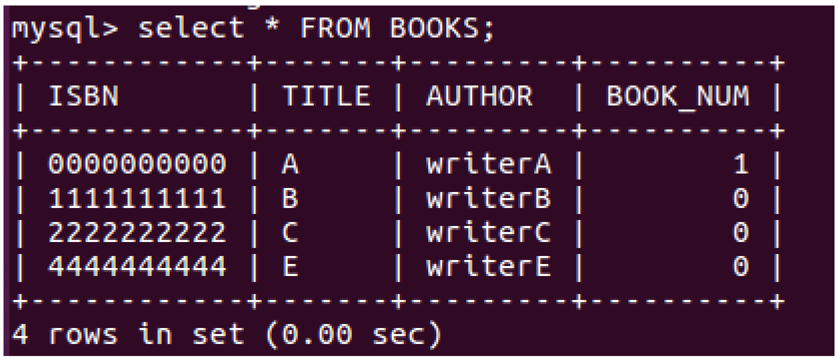
测试：

删除掉３３３３３３３３３３号书：



```
What are you going to do?
1.Add book to library
2.Remove book from library
3.Add student account
4.Query books
5.Qurey borrow history
6.Qurey unreturned book
7.Check the deadline
8.Extend the deadline
9.Check overdue
10.Suspend student accounts
0.Back to showmenu
2

Please enter ISBN
3333333333
This kind of book has been successfully removed from the library!
```

执行功能后，库的状态：



```
mysql> select * FROM BOOKS;
+------------+-------+---------+----------+
| ISBN       | TITLE | AUTHOR  | BOOK_NUM |
+------------+-------+---------+----------+
| 0000000000 | A     | writerA |        1 |
| 1111111111 | B     | writerB |        0 |
| 2222222222 | C     | writerC |        0 |
| 4444444444 | E     | writerE |        0 |
+------------+-------+---------+----------+
4 rows in set (0.00 sec)
```

至此，完成了所有功能的设计