IDBS ASSIGNMENT3: A Library System

18307130126 林梓航

1 简介

作业要求参考：

```
https://ichn-hu.github.io/IDBS-Spring20-Fudan/assignment3/readme/
```

概括一下主要是：

加书，删书，添加学生账号（管理员权限）；

根据书名，作者名或ISBN码查询书目；

使用学生账号借书，还书；

查询一个学生的借书记录；

查询一个学生借阅未还的所有书目；

检索某一本书的应还日期；

延长某一本书的应还日期（最多三次）；

查询一个学生有多少逾期未还的借阅书籍；

若学生有多于三本逾期未还的借阅书籍，则在其归还至逾期未还的借阅书籍不多于三本之前，暂停其借阅书籍的权限。

项目代码仓：

```
https://github.com/EZlzh/IDBS-ass3
```

（附有README，内含一些运行说明）

实验环境与须知：

实验环境

```
ProductName:  Mac OS X
ProductVersion: 10.15.4
BuildVersion: 19E287
MySQL: Server version: 8.0.19 MySQL Community Server - GPL
GO: go version go1.14.1 darwin/amd64
```

在Linux下也可运行。

go.mod:

```
module github.com/ichn-hu/IDBS-Spring20-Fudan/assignments/ass3/boilerplate

go 1.14

require (
  github.com/go-sql-driver/mysql v1.5.0
  github.com/jmoiron/sqlx v1.2.0
)
```

可以根据实际情况修改go-sql-driver或sqlx的版本。

library.go:

```
const (
  User     = "root"
  Password = "123456"
  DBName   = "ass3"
)
```

可以根据本机mysql实际情况修改相应用户名与密码。

运行实验代码：

1. 首先要在本地mysql创建一个与DBName对应的数据库。
2. 然后在代码根目录下在终端输入命令：

   1) 运行代码：

   ```
   go run library.go
   ```

   然后可以得到如下信息：

   ```
   Welcome to the Library Management System!
   Please Select User Mode: (input number)
   1: Student; 2: Administrator; 0: Exit.
   ```

   (我们在library.go中默认是一个初始化的数据库，仅保留一个ADMIN账户，后面会具体说明)

2）测试代码：

```
go test
```

然后可以得到如下信息：

```
Successfully added the book.
...
...
...
PASS
ok    github.com/ichn-hu/IDBS-Spring20-Fudan/assignments/ass3/boilerplate
x.xxxs
```

具体的测试情况在第4部分会展示。

2 关系表设计

1. 书籍表 BOOKS(ISBN, author, title, total, avail)

   ISBN 书号；author 作者; title 书名; total 馆藏该书籍总数; avail 剩余可借阅数。

   ```
   BOOKS(ISBN char(32) NOT NULL, author char(32), title char(100), total int,
   avail int, PRIMARY KEY(ISBN))
   ```

2. 学生表 STUS(UID, password)

   UID 学号；password 密码。

   ```
   STUS(UID char(32) NOT NULL, password char(32) NOT NULL, PRIMARY KEY(UID))
   ```

3. 管理员表 ADMINS(UID, password)

   UID 工号；password 密码。

   ```
   ADMINS(UID char(32) NOT NULL, password char(32) NOT NULL, PRIMARY
   KEY(UID))
   ```

4. 删书记录表 DELETE_REC(REC, ISBN, explanation, delete_date)

   REC 删书记录索引，即该条记录是库中第几条删书记录；ISBN 书号；

   explanation 删除原因；delete_date 删除日期。

   ```
   DELETE_REC(REC int NOT NULL, ISBN char(32) NOT NULL, explanation char(100)
   NOT NULL, delete_date DATE NOT NULL, PRIMARY KEY(REC))
   ```

5. 借书记录表 BORROW_REC(REC, UID, ISBN, start, exp, ret, EXtimes)

REC 借书记录索引，即该条记录是库中第几条借书记录；UID 学号；ISBN 书号；start 借书起始日期；

exp 应还日期；ret 实际归还日期；EXtimes 延期次数。

```
BORROW_REC(REC int NOT NULL, UID char(32) NOT NULL, ISBN char(32) NOT NULL, start DATE NOT NULL, exp DATE, ret DATE, EXtimes int, PRIMARY KEY(REC), FOREIGN KEY(UID) REFERENCES STUS(UID), FOREIGN KEY(ISBN) REFERENCES BOOKS(ISBN))
```

在考虑到可能在同一天会因为同种原因而丢掉ISBN号相同的书两本；并且学生可能会在一天内借阅或归还ISBN号相同的书多次。这些极端情况的产生，让我们在删书记录表与借书记录表中引入REC索引，则每一条记录都有所区分，更好地保持了数据库的完整性。

3 功能实现

1. 建表：

```
func (lib *Library) CreateTables()
```

2. 加书：

```
func (lib *Library) AddBook(title, author, ISBN string)
```

3. 删书：

```
func (lib *Library) DeleteBook(ISBN string, explanation string)
```

4. 添加学生账户：

```
func (lib *Library) AddStudent(UID, code string)
```

5. 根据某一信息查询相应书籍：

```
func (lib *Library) QueryBook(value, key string)
```

6. 学生借书：

```
func (lib *Library) BorrowBook(UID, ISBN string)
```

7. 查询某学生借书记录：

```
func (lib *Library) QueryHistory(UID string)
```

8. 查询某学生未归还的书：

```
func (lib *Library) QueryBooksNotReturned(UID string)
```

9. 查询ISBN号对应某书的归还日期：（可能有多本，以便借书）

```
func (lib *Library) QueryDueDate(UID, ISBN string)
```

10. 延长归还日期：

```
func (lib *Library) ExtendDueDate(REC int, UID, ISBN string)
```

11. 检查某学生是否有逾期未还的书：

```
func (lib *Library) QueryBooksOverdued(UID string)
```

12. 用学生账户还书：

```
func (lib *Library) ReturnBook(REC int, UID,ISBN string)
```

13. 暂停学生账户借书：(这里直接在借书的函数中进行限制，无需设计新的表属性或借口)

```
s1 := fmt.Sprintf("SELECT COUNT(*) FROM BORROW_REC WHERE UID = '%s' AND
exp<CURRENT_DATE() AND ISNULL(ret) ", UID)
rows, err := lib.db.Query(s1)
if rows.Next() {
      err := rows.Scan(&count)
      if err != nil {
        panic(err)
      }
    } else {
      count = 0
    }
    if count > 3 {
      fmt.Println("Account suspended. Please return books first.")
```

撰写报告时发现原来要求是有严格大于三本逾期未还的书才进行限制，在编程时考虑的是大于等于三本，不过原理是一样的。

上述功能具体实现可在代码仓进行查看。

4 测试情况

部分测试表格展示：

```
mysql> select * from BOOKS;
+----------+----------+---------------------------+-------+-------+
| ISBN     | author   | title                     | total | avail |
+----------+----------+---------------------------+-------+-------+
| 1234-5-6 | Peipei   | Cheerful_And_Humorous_Talk |     9 |     9 |
| 3690-5-6 | QWQ      | Bin_Dog                   |     1 |     0 |
| 5678-5-6 | Daye Xue | Bin_Cat                   |     4 |     1 |
| 9570-5-6 | Alpha    | How to Debug              |     1 |     0 |
| 9999-5-6 | Xiao as  | DST_master                |    10 |    10 |
+----------+----------+---------------------------+-------+-------+
5 rows in set (0.00 sec)


mysql> select * from STUS;
+-------------+----------+
| UID         | password |
+-------------+----------+
| 16302345678 | 123321   |
| 18306666333 | 123456   |
| 18307777777 | 123456   |
+-------------+----------+
3 rows in set (0.00 sec)


mysql> select * from ADMINS;
+------+----------+
| UID  | password |
+------+----------+
| root | 123456   |
+------+----------+
1 row in set (0.00 sec)


mysql> select * from DELETE_REC;
+-----+----------+-----------------------------------------------------+-------
------+
| REC | ISBN     | explanation                                         |
delete_date |
+-----+----------+-----------------------------------------------------+-------
------+
|   1 | 1234-5-6 | When Peipei was playing magic, he burnt this book. | 2020-
05-08  |
+-----+----------+-----------------------------------------------------+-------
------+
1 row in set (0.00 sec)


mysql> select * from BORROW_REC;
```

```
+-----+-------------+----------+------------+------------+------------+-------
--+
| REC | UID         | ISBN     | start      | exp        | ret        |
EXtimes |
+-----+-------------+----------+------------+------------+------------+-------
--+
|   1 | 16302345678 | 5678-5-6 | 2020-02-01 | 2020-04-30 | NULL       |
3 |
|   2 | 16302345678 | 3690-5-6 | 2020-03-01 | 2020-04-15 | NULL       |
0 |
|   3 | 16302345678 | 9570-5-6 | 2020-02-28 | 2020-04-29 | NULL       |
0 |
|   4 | 18307777777 | 5678-5-6 | 2020-05-08 | 2020-07-21 | 2020-05-08 |
1 |
|   5 | 18307777777 | 5678-5-6 | 2020-05-08 | 2020-07-07 | NULL       |
0 |
+-----+-------------+----------+------------+------------+------------+-------
--+
5 rows in set (0.00 sec)
```

我们可以看到本数据库能比较好地解决一个学生在一天内借阅多本书号相同的书的情况，并且归还其中一本时对于归还情况有所区分。

因为考虑到如何区分归还的书对应的记录，因此学生在查询借书记录时，我们会给出相应借书记录索引REC，学生需要根据该索引来指定他想要归还的书对应哪条借书记录。这样的话学生可以自由选择归还期限临近的书，以此来填充该条记录的实际归还日期。（当然也可以期限较远的书先归还，比如上面UID=18307777777, ISBN=5678-5-6, start=2020-05-08的例子）

测试输出：

我们的测试输出是交互时每个函数操作相对应的输出，具有良好的终端交互性。

```
Successfully added the book.
Successfully added the book.
Successfully added the book.
Successfully added the book.
Successfully added the book.
Successfully added the book.
Successfully added the book.
Successfully added the book.
Successfully added the book.
Successfully added the book.
Successfully added the book.
Successfully added the book.
Successfully added the book.
Successfully added the book.
```

```
Successfully added the book.
Successfully added the book.
Successfully added the book.
Successfully added the book.
Successfully added the book.
Successfully added the book.
Successfully deleted the book.
Can't find this book.
Successfully added the student!
Error: UID already exists!
Successfully added the student!
Error: UID already exists!
Find books No.1: ISBN=9999-5-6 author=Xiao as title=DST_master total=10
avail=10
Find books No.1: ISBN=1234-5-6 author=Peipei title=Cheerful_And_Humorous_Talk
total=9 avail=9
UID error!
No book is available now.
ISBN error! Can't find the book.
Successfully borrow the book!
Successfully borrow the book!
Account suspended. Please return books first.
Find borrow_records No.1: UID=16302345678 ISBN=5678-5-6 start=2020-02-01
expected=2020-04-30 return=NULL Ext_times=0
Find borrow_records No.2: UID=16302345678 ISBN=3690-5-6 start=2020-03-01
expected=2020-04-15 return=NULL Ext_times=0
Find borrow_records No.3: UID=16302345678 ISBN=9570-5-6 start=2020-02-28
expected=2020-04-29 return=NULL Ext_times=0
Find books not returned No.1: UID=16302345678 ISBN=5678-5-6 start=2020-02-01
expected=2020-04-30 return=NULL
Find books not returned No.2: UID=16302345678 ISBN=3690-5-6 start=2020-03-01
expected=2020-04-15 return=NULL
Find books not returned No.3: UID=16302345678 ISBN=9570-5-6 start=2020-02-28
expected=2020-04-29 return=NULL
The deadline of borrow_records No.1: ISBN=5678-5-6 expected=2020-04-30
Successfully extend the due!
Can't extend the due!
No book is overdued by UID 18307777777.
Find books overdued No.1: UID=16302345678 ISBN=5678-5-6 expected=2020-04-30
Find books overdued No.2: UID=16302345678 ISBN=3690-5-6 expected=2020-04-15
Find books overdued No.3: UID=16302345678 ISBN=9570-5-6 expected=2020-04-29
Successfully returned the book!
No such a book borrowed by REC=1, UID=16302345678 ISBN=5678-5-5!
PASS
ok    github.com/ichn-hu/IDBS-Spring20-Fudan/assignments/ass3/boilerplate
0.187s
```

5 数据库特点

总体来说，本数据库考虑了多种极端情况，比如考虑到可能在同一天会因为同种原因而丢掉ISBN号相同的书两本；并且学生可能会在一天内借阅或归还ISBN号相同的书多次。这些极端情况的产生，让我们在删书记录表与借书记录表中引入REC索引，则每一条记录都有所区分，更好地保持了数据库的完整性。于此同时，本数据库能比较好地解决一个学生在一天内借阅多本书号相同的书的情况，并且归还其中一本时对于归还记录有所区分，更有利于学生借阅同类书籍时合理选择，归还较早的一本。

6 参考资料

[1] https://blog.alexellis.io/golang-writing-unit-tests/

[2] https://gobyexample.com/testing

[3] https://labix.org/gocheck

[4] https://godoc.org/github.com/jmoiron/sqlx

[5] https://blog.csdn.net/KingEasternSun/article/details/78262528

7 总结与致谢

本次实验让我更好地理解了如何更好地让Golang操作数据库，同时也对Golang有了更深的理解，比如我对于其特殊的if...else...的缩进风格深有感触。

本次实验我也更好地理解了在设计一个数据库系统中所要考虑的方方面面，在设计模型时怎样考虑到各种复杂情况，可能还有些不足，但是为我今后的学习工作生活打下了基础。

感谢老师助教与同学们在我实验中的帮助。