IDBS Ass3 library 报告

贾子安 18307130017

1. 简介

- 简单的图书管理系统
- 项目仓库：https://github.com/Jimmyokok/Tiny-Library
- 使用Go语言编写，数据库采用MySQL

2.数据库模式

- 图书馆数据库(library)包含以下模式：
  - admin(主键id, name, password)
  - student(主键id, name, password)
  - book(主键ISBN, author, title)
  - bookcount(外键ISBN, count)
  - active_record(外键stuid, 外键bookid, due, exttimes)
  - history_record(外键stuid, 外键bookid, end)
  - delete_record(自增主键id, ISBN, reason)
- `admin`表记录所有图书馆系统的管理员账号，每个管理员的id都不相同，登录时需要验证password是否输入正确。
- `student`表记录图书馆系统登记的所有学生，学生各自的id都不相同，但是和管理员可以相同，登录时同样需要验证password。
- 由于ISBN唯一决定一本书的属性，而图书馆内又经常会同时存在很多本同样的书，存在书籍信息冗余，因此我将存储书籍信息的模式拆分成了两张表。其中`book`表存放某一ISBN对应的书的所有信息，ISBN不会重复，`bookcount`表存放该ISBN所对应的书的库存数量。
- `active_record`存放目前仍然没有归还的借书记录，其中stuid是学生学号，bookid是书的ISBN号，due是归还的截止日期，exttimes是该借书记录的归还期限已经被延长的次数。
- `history_record`存放已经归还的借书记录，其中end是归还的日期，其余同上。
- `delete_record`存放删除书籍的记录，由于每次只能删除一本书籍，因此删除记录中的ISBN号会发生重复，因此使用一个自增的索引来唯一代表某一个删除记录。

3. 实现功能

- 图书馆基本功能
  - 用户管理(管理员，学生)
  - 管理员和学生用户之间区别权限
  - 管理员可以添加其他管理员和学生账号
  - 增加/删除书籍
  - 借书/还书
  - 学生现存三次以上逾期不还的书则不能继续借书
  - 管理员可以延长学生的借书期限
  - 查询书籍和借书记录
  - 管理员可以查询所有账号(不包括密码)
- 交互界面

    - 命令行界面
    - 需要先登录
    - 根据登录的身份不同可以执行不同的操作
- test
    - 除了最后一个test函数，其他test函数检测各基本功能的正确与否，目前是全部通过。
    - 最后一个test函数负责测试这些基本功能的效率。

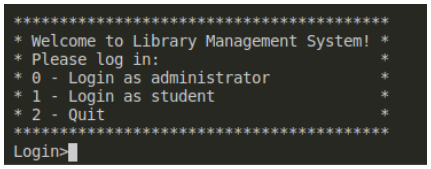
4. 实现方式

- 基本功能：采用Go函数实现以上功能，在函数里操作sql语句。

  比如添加一本书：

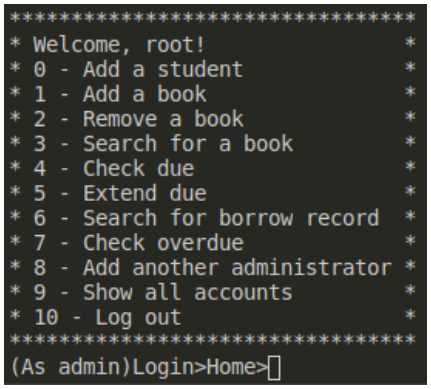
```
func (lib *Library) AddBook(title, author, ISBN string) error {
  /*
  ADMINISTRATORS ONLY
  add a book into the library
  if exists add to stock
  */
  var message string
  row := lib.db.QueryRow(fmt.Sprintf("SELECT count FROM bookcount WHERE ISBN
= '%s';",ISBN))
  var count int
  err := row.Scan(&count)
  if err == nil{//There is such a book
    lib.db.Exec(fmt.Sprintf("UPDATE bookcount SET count = count + 1 WHERE
ISBN = '%s';",ISBN))
    message=fmt.Sprintf("Administrator added book %s to stock.",ISBN)
    writelog(message)
  }else {//There is no such book
    lib.db.Exec(fmt.Sprintf("INSERT INTO book (ISBN,author,title) VALUES
('%s','%s','%s');",ISBN,author,title))
    lib.db.Exec(fmt.Sprintf("INSERT INTO bookcount (ISBN,count) VALUES
('%s',%d);",ISBN,1))
    message=fmt.Sprintf("Administrator added book %s.",ISBN)
    writelog(message)
  }
  message_board(message,[]string{})
  return nil
}
```

- 交互界面：
    - 通过Messagebox与用户交互，Messagebox形如：



    - 通过一个函数`func message_board(message string,strs []string)`实现消息框，该函数接受标题(message)和内容(strs)，输出由*包裹的消息。

- 用户通过类似以下的消息框来执行各种操作：



  - 函数`func operation(id string,is_admin bool,lib *Library)`将会负责处理用户的操作。
  - 假如用户不登出或者是退出系统，在用户执行完任意操作后会回到起始位置，比如输入操作(已登录)和登录(未登录)

- test

  - test函数先执行一遍某基本操作，然后将操作后的数据库表内内容和应有的答案相比较，如果有任意不同则不通过。
  - 例如：

```
func TestCreateTables(t *testing.T) {
  var ans_tablename = []string{
    "active_record",
    "admin",
    "book",
    "bookcount",
    "delete_record",
    "history_record",
    "student",
  }//应有的结果
  lib := Library{}
  lib.ConnectDB()
  lib.DeteleTables()//删除所有图书馆数据库内的表
  err := lib.CreateTables()
  if err != nil {
    t.Errorf("Testing CREATETABLE:[Error]Unknown error")
  }
  //检测图书馆数据库里的所有表
  rows1,err := lib.db.Query("SHOW TABLES;")
  if err != nil{
    panic(err)
  }
  defer rows1.Close()
  var tablename string
  var i int
  for rows1.Next(){
    rows1.Scan(&tablename)
    //将实际数据和应有数据比较
    if (tablename!=ans_tablename[i]) {
      //输出错误
      t.Errorf(fmt.Sprintf("Testing CREATETABLE:[Error]line %d in
```

```
tablenames, want %s, got %s.\n",i+1,ans_tablename[i],tablename))
    }
    i++
  }
  defer lib.db.Close()
}
```

5. 使用

- 确保装有MySQL和go编译器与环境
- 确保所在文件夹拥有权限
- 使用以下指令即可打开图书馆管理系统：

```
go run library.go
```

- 默认使用的MySQL账号为：

```
User = "root"
Password = "123456"
```

可以在library.go中更改

- 由于该系统中只有管理员能添加其他管理员和学生账号，因此第一次打开该系统时系统会自动生成一个管理员：

```
(id, name, password) = ("0", "root" ,"123456")
```

- 使用以下指令即可运行测试程序：

```
go test -v
```

样例结果如下：

```
=== RUN   TestCreateTables
--- PASS: TestCreateTables (0.10s)
=== RUN   TestAddBook
--- PASS: TestAddBook (0.13s)
=== RUN   TestRemoveBook
--- PASS: TestRemoveBook (0.03s)
=== RUN   TestQueryBookbyISBN
--- PASS: TestQueryBookbyISBN (0.00s)
=== RUN   TestQueryBookbytitle
--- PASS: TestQueryBookbytitle (0.00s)
=== RUN   TestQueryBookbyauthor
--- PASS: TestQueryBookbyauthor (0.00s)
=== RUN   TestAddStudent
--- PASS: TestAddStudent (0.02s)
=== RUN   TestBorrow
--- PASS: TestBorrow (0.08s)
=== RUN   TestCheckDue
--- PASS: TestCheckDue (0.00s)
=== RUN   TestExtendDue
--- PASS: TestExtendDue (0.01s)
=== RUN   TestCheckOverdue
--- PASS: TestCheckOverdue (0.00s)
=== RUN   TestReturnBook
--- PASS: TestReturnBook (0.03s)
=== RUN   TestQueryBorrowHistory
--- PASS: TestQueryBorrowHistory (0.00s)
=== RUN   TestQueryBorrowCurrent
--- PASS: TestQueryBorrowCurrent (0.00s)
=== RUN   TestEfficiency
--- PASS: TestEfficiency (17.78s)
=== RUN   TestDeteleTables
--- PASS: TestDeteleTables (0.05s)
PASS
ok      library 18.247s
```