

Ass3 实验报告

18307130116 王博

1.test 文件截图

```
musicode@ubuntu:~/Desktop/repo/IDBS-Spring20-Fudan/assignments/ass3/submission/18307130116$ go test
DONE
DONE
DONE
{book1 li 100 borrowed}
DONE
{book5 a5 500 ready}
DONE
{book6 a6 600 ready}
DONE
{book5 a5 500 ready}
DONE
{book1 li 100 borrowed}
DONE
{book3 zhang 300 borrowed}
DONE
Error! This book has been borrowed by others
DONE
{2020-04-16 13:33:04 100 student3 borrow 2020-04-30 13:33:04}
{2020-04-17 13:33:04 100 student3 return 2020-04-17 13:33:04}
No information
No information
{book3 zhang 300 borrowed}
2020-04-30 13:33:04
2020-05-19 13:33:04
You have 2 times left
You have 1 times left
DONE
PASS
```

由于在最初设计的过程中，我的目标是一个类似于 cmd 命令行的工具，于是在 main 函数内其实完成的工作就是链接初始化，以及根据相应的 command 来执行相应的功能，每个功能所需要的输出是分布在了各个 function 里面，因此在 test 的过程中，仍然会进行输出

2.文件

在学号文件夹下共有两个子文件夹，library 文件夹里面放置的是主程序（library.go）和测试程序（library_test.go），support 文件夹里包括三个文件 drop_table.go, create_table.go, insert_table.go,这三个文件分别对应了删除表格，创建表格和插入表格三个功能

3.使用该系统

<https://github.com/obangw/Library>

(1) 正常使用

①在本地 sql 配置 ass3 数据库，用户 root，密码为 123（也可以更改 library 中的 const 值）

②运行 support 中的 create_table.go，完成创建基本表操作，默认添加了 root，密码为 123，可像上一步一样更改 const 值

③完成上述操作之后即可运行 library 文件夹中的 library.go，help 命令将会调出命令列表，正常使用即可，**可以通过命令的方式完成正常的交互操作**

(2) 测试使用

①在本地 sql 配置 ass3 数据库，用户 root，密码为 123（也可以更改 library 中的 const 值）

②运行 support 中的 create_table.go，insert_table.go，完成创建基本表和导入数值操

作，默认添加了 root，密码为 123，可像上一步一样更改 const 值

③此时的插入的 time 数值基于了本地 os 的时间，必须调用 mysql 命令行，查询此时的 HISTORY 基本表，更改 test 文件中的 TIME1 和 TIME2 常量为图片标注的两个对应数值（即 student3 第一次 borrow 的 due 和 student1 第三次 extend 的 due）

time	ISBN	user	op	due
2020-04-02 13:33:04	200	student1	borrow	2020-04-16 13:33:04
2020-04-11 13:33:04	400	student2	borrow	2020-04-25 13:33:04
2020-04-12 13:33:04	200	student1	return	2020-04-12 13:33:04
2020-04-16 13:33:04	100	student3	borrow	2020-04-30 13:33:04
2020-04-17 13:33:04	100	student3	return	2020-04-17 13:33:04
2020-04-18 13:33:04	100	student2	borrow	2020-05-02 13:33:04
2020-04-20 13:33:04	200	student2	borrow	2020-05-04 13:33:04
2020-04-21 13:33:04	400	student2	extend	2020-05-05 13:33:04
2020-05-02 13:33:04	300	student1	borrow	2020-05-16 13:33:04
2020-05-03 13:33:04	300	student1	extend	2020-05-17 13:33:04
2020-05-04 13:33:04	300	student1	extend	2020-05-18 13:33:04
2020-05-05 13:33:04	300	student1	extend	2020-05-19 13:33:04
2020-05-07 06:35:02	500	other	borrow	2020-05-21 06:35:02
2020-05-07 06:35:03	600	other	borrow	2020-05-07 06:35:03
2020-05-07 06:35:06	200	student2	extend	2020-05-21 06:35:06
2020-05-07 06:35:06	400	student2	extend	2020-05-21 06:35:06
2020-05-07 06:35:07	100	student2	return	2020-05-07 06:35:07

17 rows in set (0.00 sec)

④更改完成后进入 library 目录下，运行 gotest 进行测试

⑤重复测试需要运行 drop_table.go 后重复②开始的操作

注意：测试数据为设计好的，如果更改测试数据，由于 HISTORY 的主键为 time，user，ISBN，time 精确级别为秒，可能出现修改了某个元组后，insert 了一个同时同账户同书本的元组导致 insert 出错，但那只是因为 test 中操作速度显著高于判断精度，实际考虑中，有理由认为一个人在一个时间（秒级别）对一本书只会有一个操作，因此不会在现实使用中出问题

4. 总体设计思路

(1) 基本表

STUDENT (user, password)

HISTORY(time, user, ISBN, op, due)

BOOK(title, author, ISBN, state)

衡量了各个操作，第一个表格只需要 user 和 password 两个值，并且主键设置为 user，因为用户名肯定需要保证不重复，因为有查询历史纪录的操作，因此最好是对 history 只添加，不修改，根据后续函数要求，设置了如上五个变量，并且希望一些操作可以精确到秒，这就要求时间不由用户输入，且时间对应的数据类型为 datetime 类型，考虑到后续 test 过程中，各个 test 函数其实是几乎同时进行操作，仅设置主键为 time 显然无法满足要求，**我们有理由认为，一个用户在一个时间对一本书只有一个操作**，因此设置了主键为 time，user，ISBN，而 book 基本表较为简单，state 的存在确实会造成少量的数据冗余，但是在后续判断的过程中将会提供极大地便利，因此添加了 state 属性，**并且根据常**

识知道，ISBN 与书是一一对应关系，所以选取了 ISBN 作为该表格的主键

(2) 解决初始化问题和 login 问题

在整个 library 的完成过程中，首先需要解决的是 user 怎么去 login，由于想做的是一个模拟器，应该保证每次打开系统，数据不会重复更新或插入，因此我将数据库初始化配置和模拟器区分开了，这就要求，为了使模拟器运行，需要单独在 mysql 命令行中创建完成 ass3 这个数据库，并且运行 create_table.go 和 insert_table.go 两个文件，完成相应的初始化。当然 insert_table.go 只是为了测试方便导入了一些数据，单纯的使用系统只需运行 create_table.go 即可，会默认创建 root 账户，密码为 123。

最初在构思整个 login 框架的时候，其实想利用 connectDB，完成 login 操作，即，在 library 里面创建了一个新的账户等价于创建了一个 mysql 账户，并且给出相应的权限限定。但是这个操作过于复杂，退而求其次，我们在数据库中存放了一个学生基本表，每次都使用 root 账户和数据库之间进行 connect，查询相应的 password 并加以比对，从而完成了一个伪的 login 过程，对应的 addaccount 函数也就有了思路

(3) 为实现函数更改 lib 类

部分函数需要利用到当前账户中到底有多少本超期未还的书目，因此在登录最初期就将书目的数量赋给了 lib 类的 overdue_num 分量，user 分量则记录的是当前 login 的账户是谁，便于在后续过程中调用，overdue_num 将会在每个 command 结束时运算一遍，以确保相应数值的更新

5.函数及对应的测试思路

(1) ConnectDB:

完成的工作是与数据库相连接，从而开展后续的一系列操作

测试思路：直接通过返回值 err，判断是否完成了连接操作

(2) AddBook:

完成的作为向 BOOK 表格中添加书目，并且设定对应的 state 为 ready，在添加书目之前，首先在 main 函数里面完成了对当前用户的判断，因为添加书的操作仅可由管理员为完成，如果当前用户不是管理员，那么将会输出 No Authority，并且拒绝调用该函数

测试思路：在 test 文件中查询 BOOK 表，如果没有结果那么说明添加错误，选取其中的 title 为代表属性，如果 title 和预设值的不一样，那么将会直接导致错误

(3) Addaccount:

完成的作为向 STUDENT 表格中添加用户和密码。和 addbook 一样，该功能也是仅管理员可以调用，在 main 函数里面会进行相应的判断。并且在添加用户的过程中需要对密码进行二次核验，如果密码不正确的话，账户也不会正确添加

测试思路：大致同上，通过预设值，运行之后查询表格中是否正确添加

(Search 部分，在 main 函数中会在读出了对应的命令之后给出选项，根据选

项调用函数，值得注意的是，设计过程中不认为标题作者和 ISBN 会出现空

格，空格不被允许)

(4) search_title:

完成的工作为通过 title 对相应的书目进行搜索，并且将会打印出该书目完整的信息，包括对应的状态

测试思路：通过调用函数，返回对应的值，并与预设值相对比，判断最终结果是否相同

(5) search_author:

基本内容同上，区别在于会通过 author 来进行搜索

(6) search_ISBN

基本内容同上，区别在于会通过 ISBN 搜索

(7) borrow

完成的工作为首先查询当前书的状态，如果状态不是 ready 的话，那么将不允许借用。其次，将会判断对应 lib 中 overdue 分量，如果此时的分量数值大于等于 3，那么将会拒绝借用。完成了查证，可以借用后，将会向 HISTORY 中插入一条元组，对应于现在的时间以及 ddl 设置为 14 天之后，并且设置书本的状态为被借走，返回值中 0 代表没有成功借走书本，返回值 1 代表成功借走

测试思路：通过调用函数，测试是否成功借走书，并且与相应的预设值进行对比，如果成功借走，那么将会对表格进行查询，找到最近的一次关于该书的记录，如果记录中的操作为 borrow 则测试成功

(8) check_account

在给定的函数参数中设置了一个 tag 项，当 tag = 0 的时候不输出，仅作为其他函数的辅助函数使用，而如果 tag = 1，将会对相应的结果进行输出，查找的思路为，找到 HISTORY 中记录的关于某个同学和某本书借书和延期记录，取延期 ddl 的最大值，找到所有还书记录，如果借书记录的书本不在还书记录中，并且当前时间已经超过了还书的 ddl 最大值，那么判定超过时限，或者书本的在记录中但是由于重复借阅的缘故，该书的还书记录小于借书记录的时间，并且当前时间超过了还书的 ddl，那么判定书本超期，并且返回值最终为超过时期的书的数量

测试思路：通过调用函数，测试对应的数量是否正确，如果超期的数量正确，那么代表测试通过

(9) command_fault

单纯的输出函数，用于在 main 中处理非法的指令

(10) Remove

完成的工作为，向 HISTORY 中插入一条 op = other 的元组，并且将对应的书本移除 BOOK 表格。最开始会通过调用 make_sure 函数确保该用户确实借阅了这本书，并且完成相应的操作。

测试思路：通过调用相应的函数，检测是否会 remove 之后是否仍有这本书，并且通过查询最近一次关于该书目的记录，判定是否为 other

(11) history

完成的工作为调出所有相关用户的记录。改功能分为两个模式，如果是 administrator 调用，那么可以调出任意用户的历史操作记录，如果是学生个体用户调用，那么只会调出对应的学生的历史操作记录

测试思路：由于这个功能比较简单，只需核验返回值是否为 nil，判断是否正常返回即可

(12) remain

完成的工作为查找对应的用户还有哪些书没有还，查询思路和之前的 checkaccount 类似，找到单个用户单本书借阅记录的最大时间，以及还书记录的最大时间，如果借书记录中有书本不在还书记录中，那么就判定未还，如果当前书本有还书记录，但是还书记录早于借书记录，那么也会判定未还。

测试思路：测试对应的返回值是否为预设值，比较对应 ISBN 是否相同，如果相同，则代表查询正确

(13) checkddl

完成的工作为查询某一个用户某本书的最迟归还日期，查询思路也是找到最迟的操作为 borrow 或者 extend 的元组，并将该元组的归还日期返回，作为最终结果，在 main 函数里，这个功能也同样被我做了区分，对于管理员，可以查询任意一名同学的任意一本书的 ddl，而同学账户只能查询自己的 ddl，如果该同学其实没有借阅这本书，那么将会输出 no information

测试思路：函数会返回相应的时间，并与预设值做对比，完全相同才可通过测试，值得注意的是由于 insert 工作是单独运行的，且在 insert 中，时间的插入是基于当前系统时间，即不同时刻运行 insert_table.go 都会得到不同的时间，因此在测试文件里，定义了两个宏，TIME1 和 TIME2，实际测试的过程中，需要在 insert 完成之后，调用 mysql 命令行，将对应的两个值重新赋值，才能正确通过测试。

(14) extend

完成的工作比较简单，向 HISTORY 中插入一条操作为 extend 的元组。实际完成的过程中，首先会查找当前记录中，该用户上次借阅书目之后，对该书本到底 extend 了多少次，如果已经 extend 了三次，返回次数，如果超过了三次，那么将会直接返回，其次将会判断该用户是否已经借阅这本书，同样是调用了 makesure 函数，完成相应的判断，最后则是将对应的元组插入到 HISTORY 中

测试思路：调用 extend 之后，将会判断查询 history 中关于本书的最近的一次记录，判断其用户是否为预设值，只有用户的确为预设值“student2”才会通过测试

(15) make_sure

这个函数主要起到的辅助作用，判断该书是否被当前用户借阅，由于用在了两个其他函数内，两个其他函数测试无误，则可以说明 makesure 函数无误，没有设置相应的函数测试

(16) returnbook

完成的工作为，首先调用 makesure 函数判断是否借过本书，如果借阅过，那么将会修改 BOOK 中书本的状态，并且向 HISTORY 中插入元组

测试思路：预设书本的 ISBN，在归还操作之后，将会调用查询 HISTORY 中关于该书

最近一次归还记录，判断是否为预设用户。进一步判断书本状态是否被正确更改为 ready，从而保证正确性

(17) help

纯输出函数，方便 library 系统的使用，无测试

6.遇到的相关问题

(1) test 书写格式问题

Test 在书写的命名过程中，应该保证 Test 后紧跟的首字母大写，否则测试识别不了

(2) time 类型的调用

Time.Format 函数本质上是将对应的时间值转化为了 string 类型的时间戳，因此可以与 sql 的联系中，部分 scan 可以直接写 string 类型的变量

(3) db.Query 和 db.QueryRow

后者会在无结果的时候返回一个 sql.ErrNoRows，可以作为值为空的时候的考量，前者在完成查询操作之后，无论有多少条元组，都需要调用.Next 函数，才能从中读出数值

(4) err 类型的重要性

对于每一个操作 sql 的语句，都应该对返回的 err 进行判断，并且加入 panic 语句，否则会出现，sql 操作并未完成，但是并不报错的情况

7.可以改进的地方

由于在最开始写函数的时候，就没有针对 test 进行书写，把输出和运算放在了一起，少了一层抽象，所以其实可以再加一层抽象，以获得运算值，简化部分 test 函数。

Insert 工作是基于当前时间的，所以有些不太方便，可以改进成基于固定时间，使得每次测试结果都相同，便于操作