数据管理系统:Bookstore

孙秋实　郑佳辰　汤琼

数据科学与工程学院 2018 级

January 13, 2021

1 数据库设计

2 基本功能设计与实现

3 进阶功能设计与实现

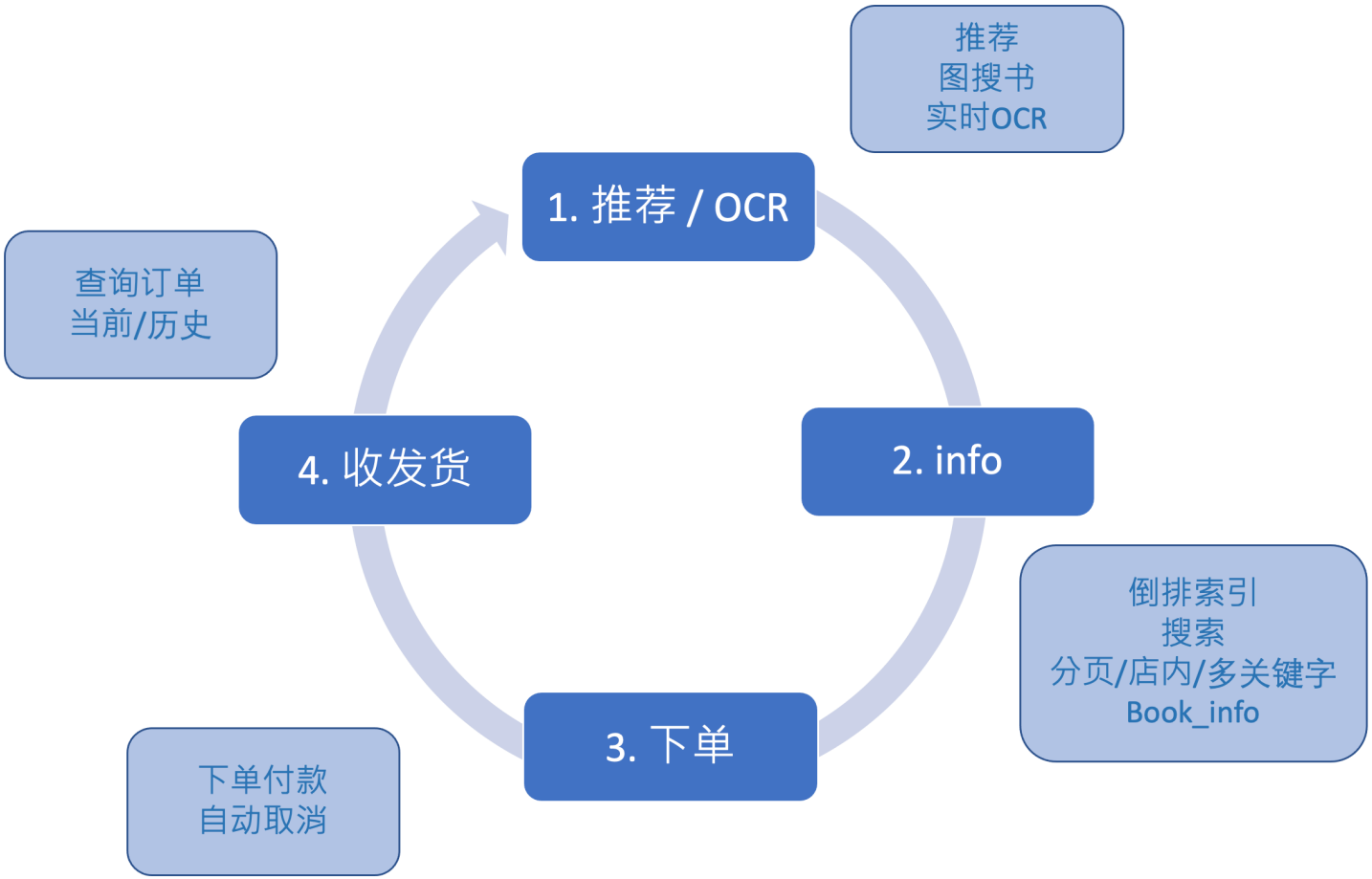
4 接口测试与性能分析

5 小组协作与分工

6 演示与提问

项目简介

A Brief Introduction



1 数据库设计
- ER 图
- 表设计与属性
- 索引设计

2 基本功能设计与实现

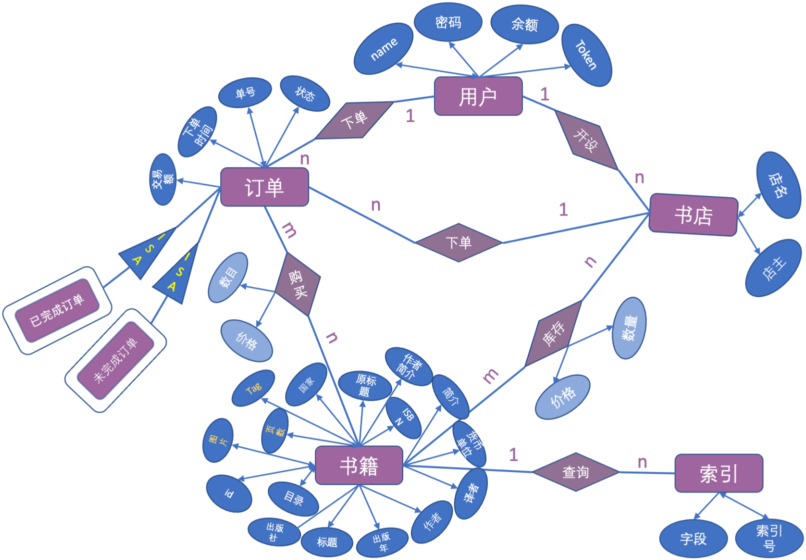
3 进阶功能设计与实现

4 接口测试与性能分析

5 小组协作与分工

6 演示与提问

ER 图



用户
name
密码
余额
Token
下单
开设
1
1
n
n
订单
单号
状态
下单时间
交易额
书店
店名
店主
下单
n
1
m
购买
数目
价格
n
ISA
ISA
已完成订单
未完成订单
n
库存
数量
价格
m
书籍
Tag
国家
原标题
作者简介
简介
ISBN
货币单位
译者
作者
出版年
标题
出版社
目录
id
图片
页数
查询
1
n
索引
字段
索引号

重要表设计： users

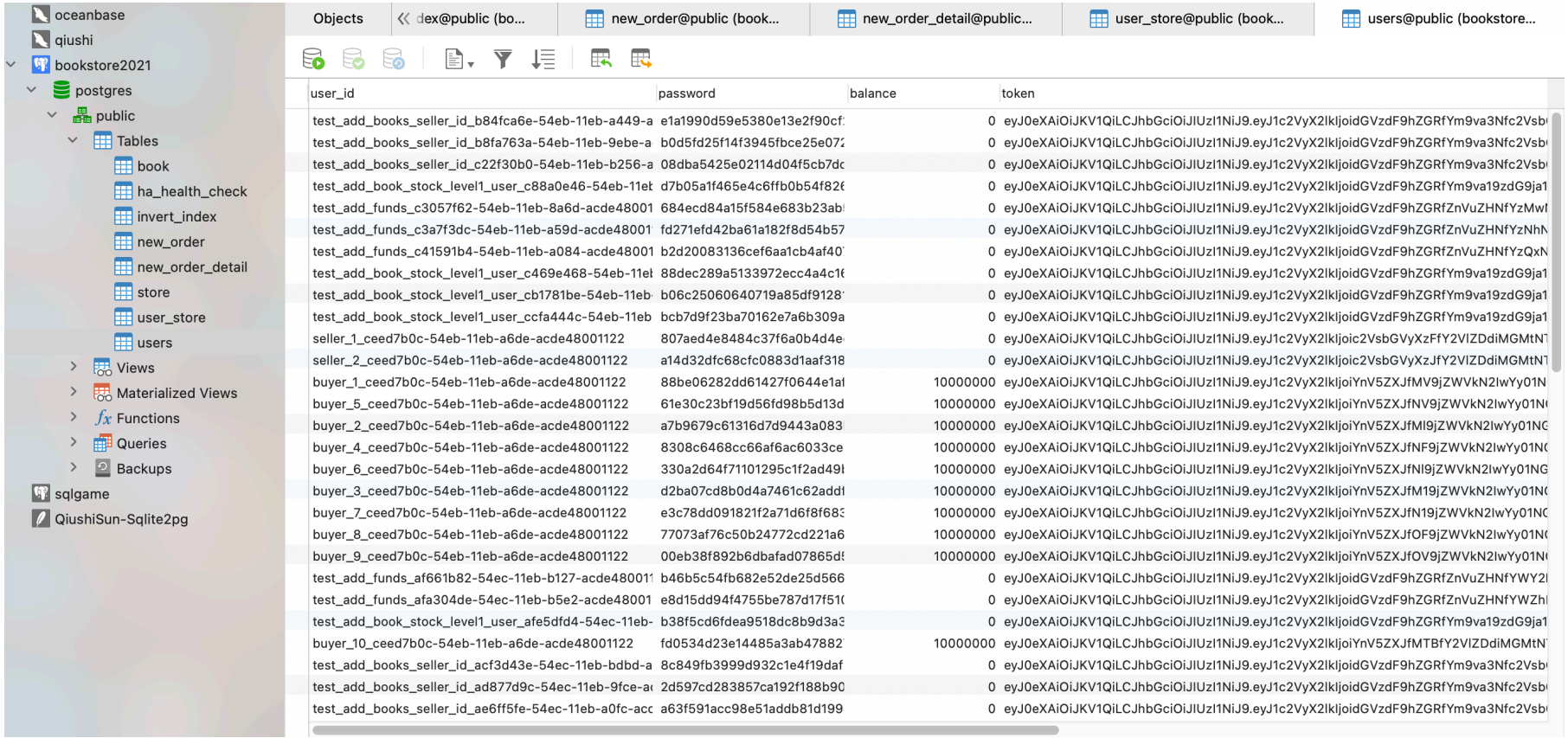


Figure: user-table

重要表设计：new-order

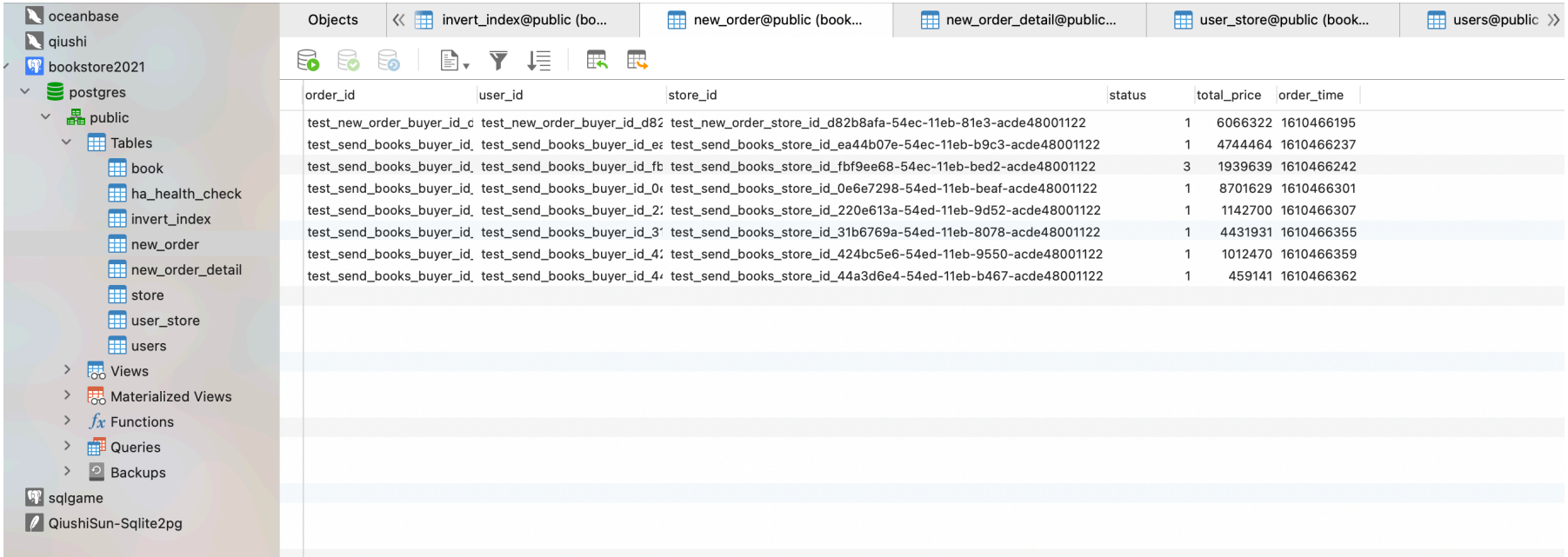
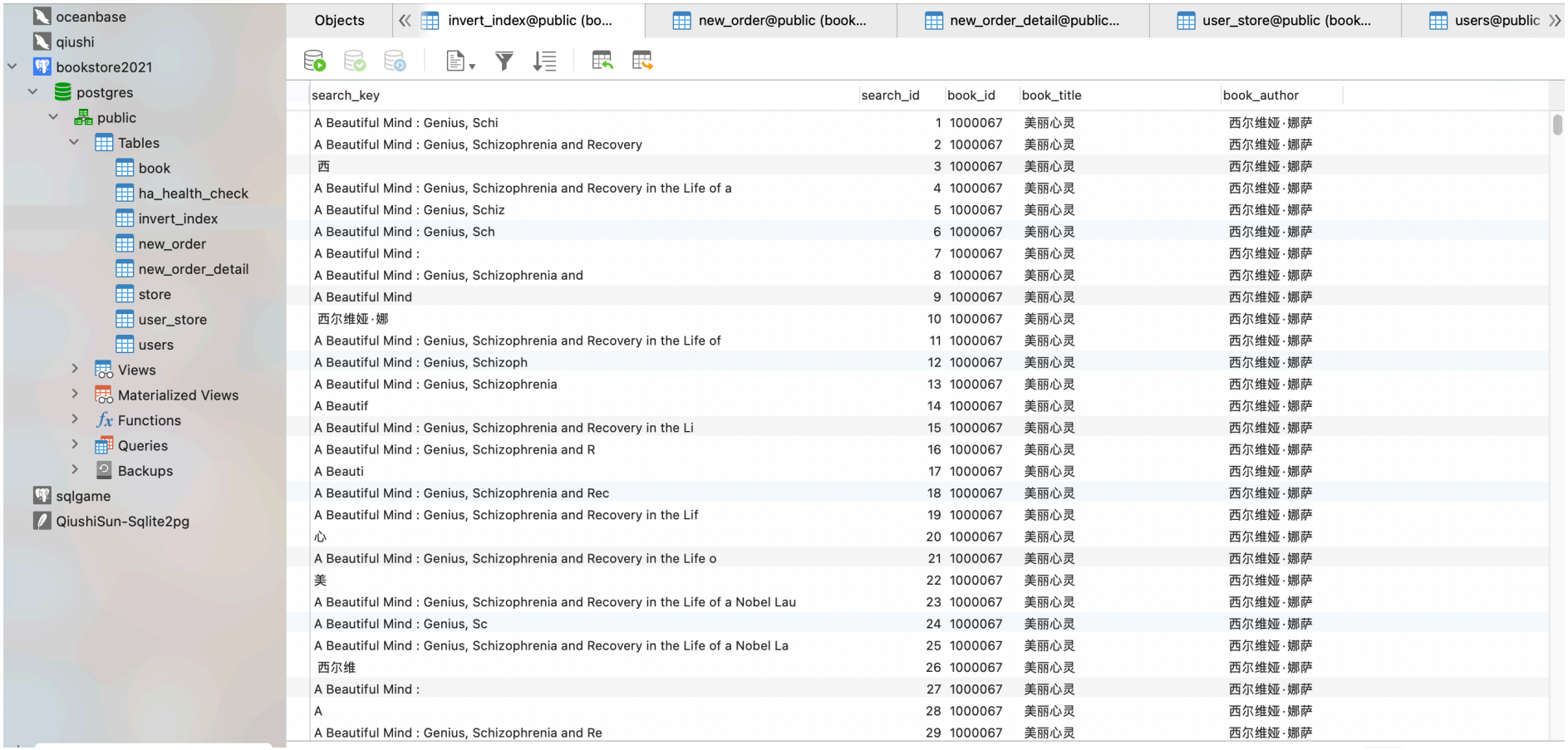


Figure: new-order

重要表设计： inverted-index



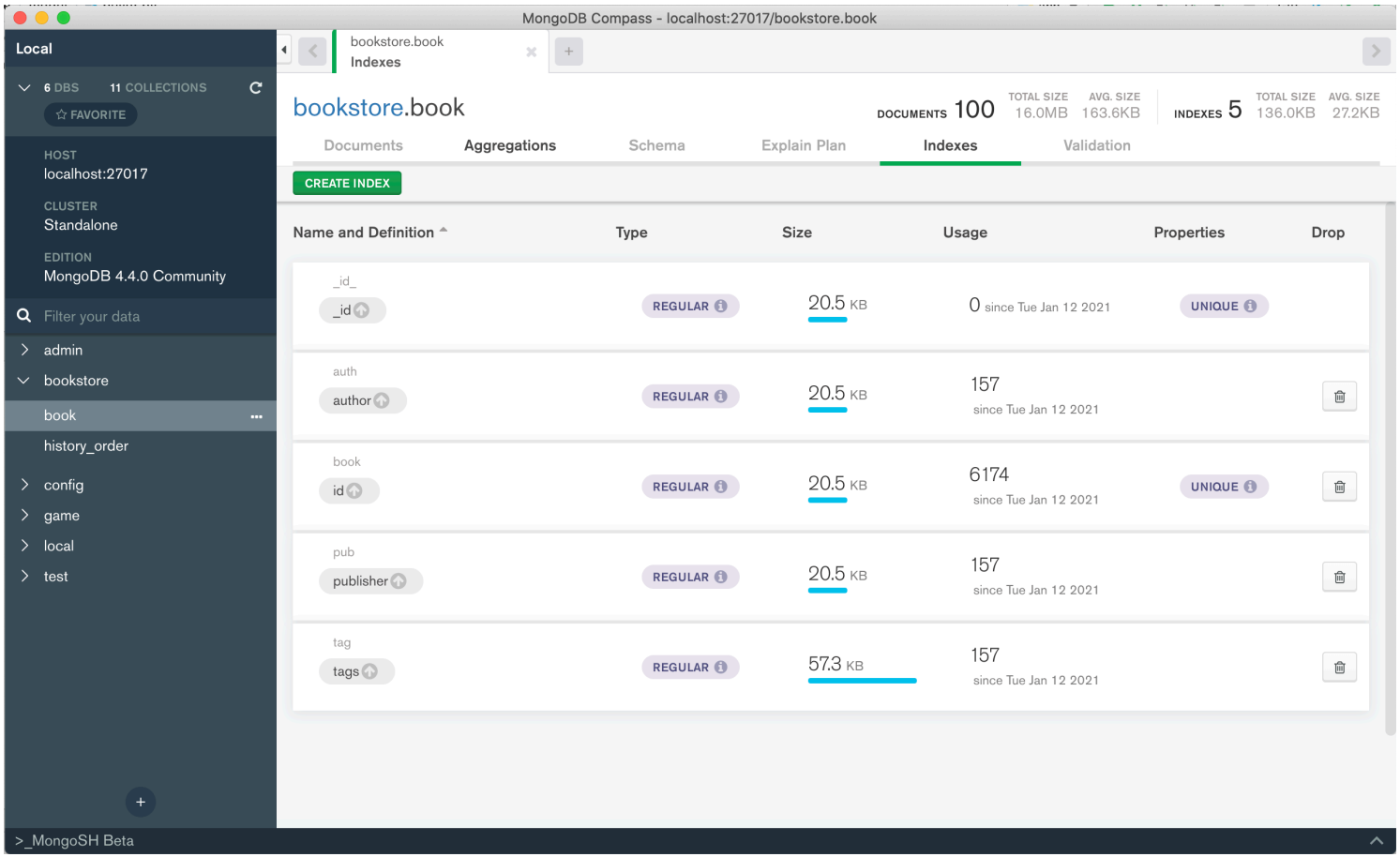
search_key	search_id	book_id	book_title	book_author
A Beautiful Mind : Genius, Schi	1	1000067	美丽心灵	西尔维娅·娜萨
A Beautiful Mind : Genius, Schizophrenia and Recovery	2	1000067	美丽心灵	西尔维娅·娜萨
西	3	1000067	美丽心灵	西尔维娅·娜萨
A Beautiful Mind : Genius, Schizophrenia and Recovery in the Life of a	4	1000067	美丽心灵	西尔维娅·娜萨
A Beautiful Mind : Genius, Schiz	5	1000067	美丽心灵	西尔维娅·娜萨
A Beautiful Mind : Genius, Sch	6	1000067	美丽心灵	西尔维娅·娜萨
A Beautiful Mind :	7	1000067	美丽心灵	西尔维娅·娜萨
A Beautiful Mind : Genius, Schizophrenia and	8	1000067	美丽心灵	西尔维娅·娜萨
A Beautiful Mind	9	1000067	美丽心灵	西尔维娅·娜萨
西尔维娅·娜	10	1000067	美丽心灵	西尔维娅·娜萨
A Beautiful Mind : Genius, Schizophrenia and Recovery in the Life of	11	1000067	美丽心灵	西尔维娅·娜萨
A Beautiful Mind : Genius, Schizoph	12	1000067	美丽心灵	西尔维娅·娜萨
A Beautiful Mind : Genius, Schizophrenia	13	1000067	美丽心灵	西尔维娅·娜萨
A Beautif	14	1000067	美丽心灵	西尔维娅·娜萨
A Beautiful Mind : Genius, Schizophrenia and Recovery in the Li	15	1000067	美丽心灵	西尔维娅·娜萨
A Beautiful Mind : Genius, Schizophrenia and R	16	1000067	美丽心灵	西尔维娅·娜萨
A Beauti	17	1000067	美丽心灵	西尔维娅·娜萨
A Beautiful Mind : Genius, Schizophrenia and Rec	18	1000067	美丽心灵	西尔维娅·娜萨
A Beautiful Mind : Genius, Schizophrenia and Recovery in the Lif	19	1000067	美丽心灵	西尔维娅·娜萨
心	20	1000067	美丽心灵	西尔维娅·娜萨
A Beautiful Mind : Genius, Schizophrenia and Recovery in the Life o	21	1000067	美丽心灵	西尔维娅·娜萨
美	22	1000067	美丽心灵	西尔维娅·娜萨
A Beautiful Mind : Genius, Schizophrenia and Recovery in the Life of a Nobel Lau	23	1000067	美丽心灵	西尔维娅·娜萨
A Beautiful Mind : Genius, Sc	24	1000067	美丽心灵	西尔维娅·娜萨
A Beautiful Mind : Genius, Schizophrenia and Recovery in the Life of a Nobel La	25	1000067	美丽心灵	西尔维娅·娜萨
西尔维	26	1000067	美丽心灵	西尔维娅·娜萨
A Beautiful Mind :	27	1000067	美丽心灵	西尔维娅·娜萨
A	28	1000067	美丽心灵	西尔维娅·娜萨
A Beautiful Mind : Genius, Schizophrenia and Re	29	1000067	美丽心灵	西尔维娅·娜萨

Figure: 倒排索引

索引设计

Mongodb-book-index (B-tree)

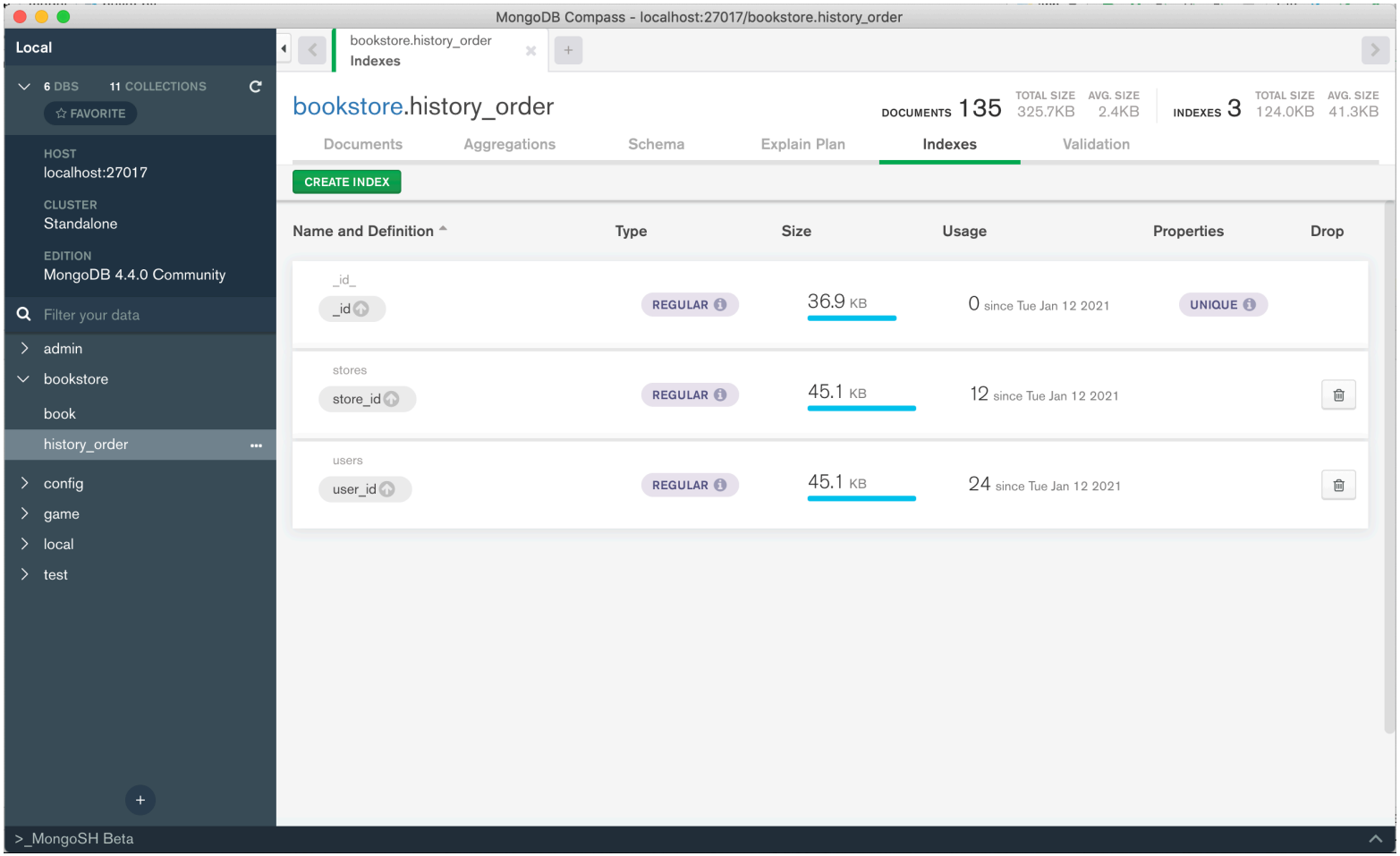
Book-index 的作用： 加速查询与推荐



索引设计 Cont'd

Mongodb-history-index (B-tree)

history-index 的作用： 加速历史信息查询（买家/卖家）



1 数据库设计

2 基本功能设计与实现
- 基本功能实现：用户权限功能
- 基本功能实现：卖家功能
- 基本功能实现：买家功能

3 进阶功能设计与实现

4 接口测试与性能分析

5 小组协作与分工

6 演示与提问

用户权限

功能实现：注册

(1) 根据 user_id 在 users 表中查询用户是否已经存在

(2) 若不存在，则插入新用户 (user_id, password, balance, token, terminal) 到 users 表中

性能: 一次查询 users table，一次插入 users table，访问数据库两次

功能实现：登陆

(1) 根据 user_id 在 users 表中获取密码 password

(2) 将获取到的 password 与用户输入的密码对比

(3) 更新用户的 token, terminal

性能: 一次查询 users table, 一次更新 users table，访问数据库两次

用户权限 Cont'd

功能实现：登出

(1) 根据 user_id 在 users table 中查询，判断登录信息是否失效

(2) 更新用户 token

性能: 一次查询 users table，一次插入 users table，访问数据库两次

功能实现：注销

(1) 根据 user_id 在 users table 中查询该用户是否存在

(2) 删除 users table 中该用户条目

性能: 一次查询 users table, 一次更新 users table，访问数据库两次

用户权限 Cont'd

功能实现：更改密码

(1) 根据 user_id 在 users table 中查询用户原先密码

(2) 判断用户原先密码和用户新密码是否相同

(3) 若不同，则更新 users table 中该用户的 password

性能: 一次查询 users table，一次插入 users table，访问数据库两次

卖家功能

功能实现：创建店铺

(1) 分别在 users table 和 store table 中查询 users_id 和 store_id 是否已经存在

(2) 若不存在，插入用户 user_id 和 store_id 到 user_store table

(3) 若不同，则更新 users table 中该用户的 password

性能: 一次查询 users table, 一次查询 store table，一次插入 user_store table，访问数据库三次

功能实现：添加库存

(1) 检查 user_id, store_id,book_id 是否存在

(2) 根据 store_id,book_id 寻找该店家某本书的库存，并在 store table 中更新

性能: 一次查询 users_table, 一次查询 user_store table, 一次查询 store table，一次更新 store table，访问数据库四次

卖家功能 Cont'd

功能实现：上架图书

(1) 检查 user_id，store_id，book_id 是否存在

(2) 根据 book_id 在 Mongodb 的 book collection 中查询 book 是否存在

(3) 根据 book_json_str 分离出作者名，作者国籍，书籍关键词，书的标题，书的作者等，插入倒排表 invert_index table

(4) (store_id, book_id, stock_level, price) 插入 store table

性能: 一次查询 users table, 一次查询 store table，一次插入 user_store table，访问数据库三次

卖家功能 Cont'd

功能实现：卖家发货

(1) 检查 store_id,book_id 是否存在

(2) 根据 order_id 在 new_order table 中更新订单状态

性能: 一次查询 user_store table, 一次查询 store table，一次更新 new_order table，访问数据库三次

买家功能

功能实现：充值

(1) 根据 user_id 获取用户密码

(2) 校验用户密码与用户输入的密码做对比

(3) 若密码一致，则更新该用户在 users table 中的余额

性能: 一次查询 users table，一次更新 users table，访问数据库两次

买家功能 Cont'd

功能实现：下单

(1) 检查 user_id, store_id 是否存在

(2) 根据订单信息（store_id，book_id）在 store 表中查找商户中是否存在对应书籍和足够的库存。

(3) 若库存足够，则更新 store table 库存

(4) 创建新订单信息，将 (order_id, book_id, count, price) 插入 new_order_detail table

(5) 创建该笔订单，计算该笔订单总价 total_price，将 (order_id, store_id, user_id, total_price, order_time) 插入 new_order table，同时将订单号 order_id 添加到 unpaid_order 数组

性能: 一次查询 users table, 一次查询 user_store table，一次查询 store table, 一次更新 store table，一次插入 new_order_detail table，一次插入 new_order table，访问数据库六次

买家功能 Cont'd

功能实现：付款

(1) 根据 order_id 在 new_order table 中查询订单信息

(2) 查询订单是否超时

(3) 若订单未超时，则根据 buyer_id 在 users table 中获取买家余额和密码

(4) 根据 store_id 在 user_store table 中查询卖家 seller_id

(5) 在 users table 中更新买家的余额

(6) 在 new_order table 中更新订单状态 status=2

性能: 一次查询 new_order table, 一次查询 users table, 一次查询 user_store table, 一次更新 users，一次更新 new_order table，访问数据库五次

买家功能 Cont'd

功能实现：收货

(1) 根据 order_id 在 new_order table 中查询对应的订单状态，买家 id

(2) 检查订单状态是否为已发货，订单 id 与买家 id 是否对应

(3) 若符合条件，则更新卖家余额，更新订单状态为已收货并且加入历史记录

性能: 一次查询 new_order table，一次查询 user_store table，两次更新 users table，一次查询 users table，访问数据库五次

买家功能 Cont'd

功能实现：取消订单

(1) 根据 order_id 在 new_order table 中查询订单状态是否为已下单

(2) 检查用户是否存在，订单号是否存在

(3) 从全局 unpaid_orders 字典中删除该订单

(4) 根据 order_id 从 new_order table 中删除该订单，从 new_order_detail table 中删除订单信息

(5) 根据 store_id 和 book_id 回滚库存 stock_level

(6) 将订单信息加入 MongoDB 的历史记录 history_order collection 中

性能: 一次查询 new_order table，一次查询 user_store table，两次更新 users table，一次查询 users table，访问数据库五次

1 数据库设计

2 基本功能设计与实现

3 进阶功能设计与实现
- 自动取消订单
- OCR 搜书
- 迷你推荐系统

4 接口测试与性能分析

5 小组协作与分工

6 演示与提问

自动订单取消

```python
from be.model.order import Order
import datetime
import time

def get_time_stamp(): #get timestamp for auto cancelling orders that are not paid in time
    cur_time_stamp = time.time()
    return int(cur_time_stamp)

time_limit = 30 # 订单存活时间
unpaid_orders = {}

#优点：通过维护全局数组to_be_paid，没有额外新启线程，代价降到最低
def add_unpaid_order(orderID):
    unpaid_orders[orderID] = get_time_stamp()
    print("add successfully")
    print(unpaid_orders)
    return 200, "ok"

def delete_unpaid_order(orderID):
    try:
        unpaid_orders.pop(orderID)
        print(unpaid_orders)
    except BaseException as e:
        return 530, "{}".format(str(e))
    return 200, "ok"
```

自动订单取消 Cont'd

```python
def check_order_time(order_time):
    cur_time = get_time_stamp()
    time_diff = cur_time - order_time
    if time_diff > time_limit:
        return False
    else:
        return True


def time_exceed_delete():
    del_temp=[]
    o = Order()
    print("new cycle start")
    for (oid,tim) in unpaid_orders.items():
        if check_order_time(tim) == False:
            del_temp.append(oid)  # remenber, not to append the index of the array, we need the orderID
    for oid in del_temp:
        delete_unpaid_order(oid)
        o.cancel_order(oid)
    return 0
```

自动订单取消 Cont'd

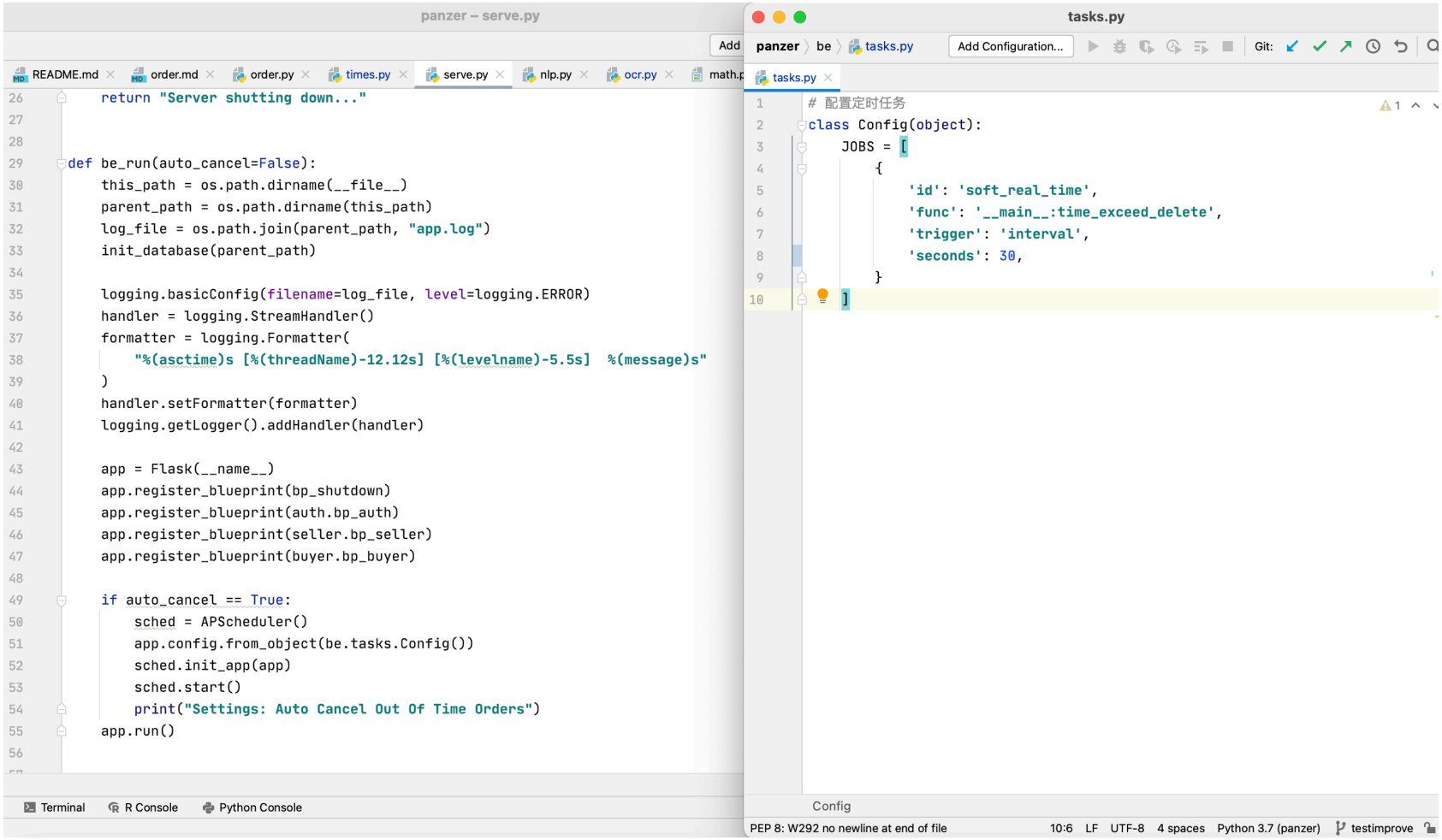


Figure: 配置定时任务

OCR 搜书

以图搜图的一种实现方法

- 为什么不用哈希方法进行图像搜索？
- 开销太大了！

OCR 搜书

以图搜图的一种实现方法

- 为什么不用哈希方法进行图像搜索？
- 开销太大了！

OCR 测试

Figure: 书本封面示例

POST http://localhost:5000/buyer/upload

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE
png	math.png
Key	Value

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "message": "ok",
    "result": [
        200,
        "ok",
        [
            {
                "author": "茆诗松,程依明,濮晓龙",
                "bid": "gll",
                "title": "概率论与数理统计教程习题与解答"
            }
        ]
    ]
}
```


Figure: Json 形式的 OCR 结果

OCR 测试 Cont'd

```
['概率论', '解答', '数理统计', '习题', '教程']
概率论
[{'bid': 'gll', 'title': '概率论与数理统计教程习题与解答', 'author': '茆诗松,程依明,濮晓龙'}, {'bid': '016855', 'title': '概率论与数理统计教程', 'author': '茆诗松,程依明,濮晓龙'}]
解答
[{'bid': 'gll', 'title': '概率论与数理统计教程习题与解答', 'author': '茆诗松,程依明,濮晓龙'}]
数理统计
[{'bid': 'gll', 'title': '概率论与数理统计教程习题与解答', 'author': '茆诗松,程依明,濮晓龙'}, {'bid': '016855', 'title': '概率论与数理统计教程', 'author': '茆诗松,程依明,濮晓龙'}]
习题
[{'bid': 'gll', 'title': '概率论与数理统计教程习题与解答', 'author': '茆诗松,程依明,濮晓龙'}]
教程
[{'bid': 'gll', 'title': '概率论与数理统计教程习题与解答', 'author': '茆诗松,程依明,濮晓龙'}, {'bid': '016855', 'title': '概率论与数理统计教程', 'author': '茆诗松,程依明,濮晓龙'}]
```

Figure: 通过 TextRank 提取词组

OCR 测试 Cont'd

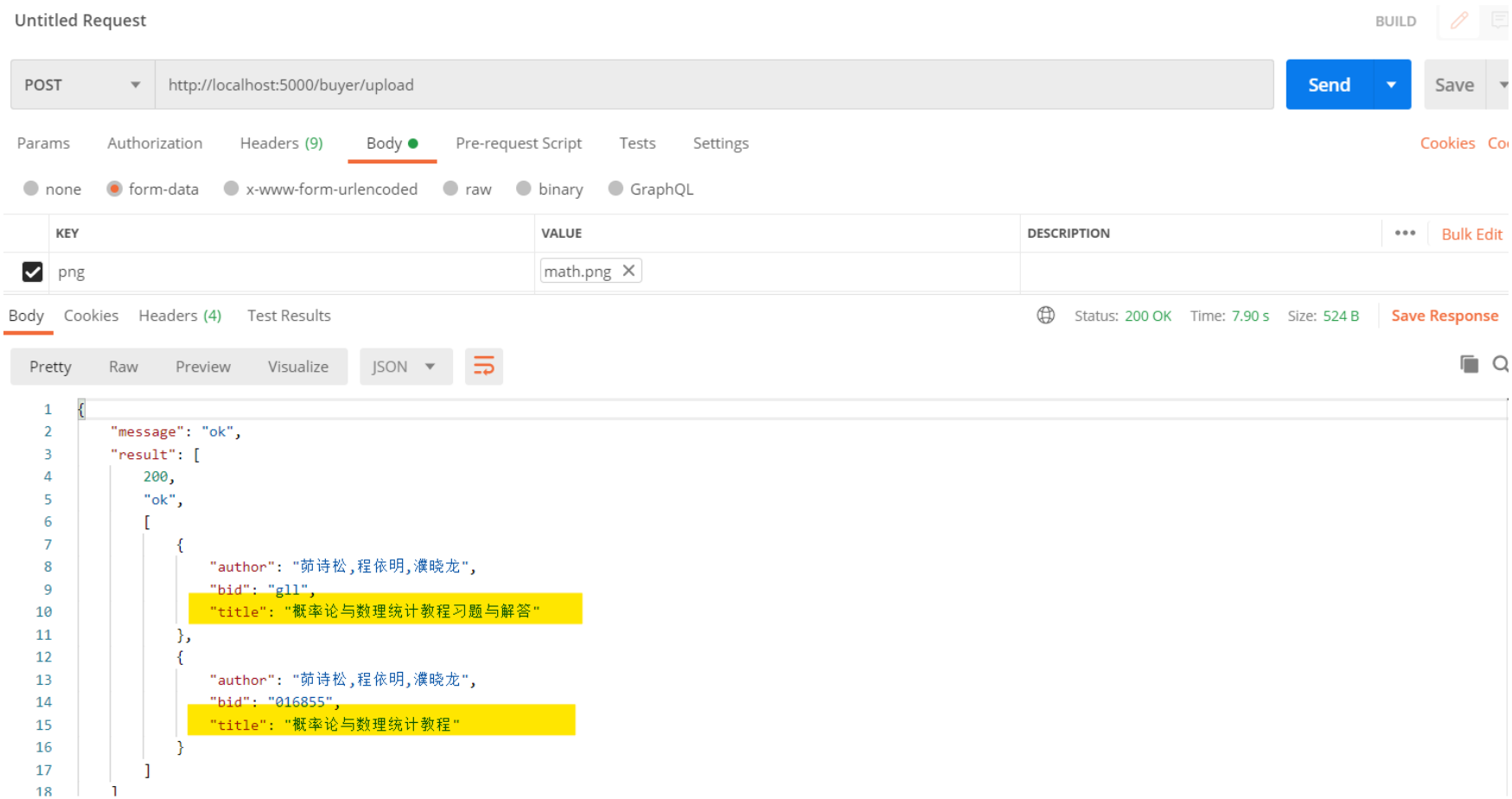


Figure: OCR 搜书结果

迷你推荐系统

```python
def recommend(self, user_id):
    try:
        code, message, orders = self.history_order(user_id)
        if code != 200:
            return error.error_non_exist_user_id(user_id)

        reco = {}
        labels = []
        for order in orders:
            boughtbooks = order['books']
            for boughtbook in boughtbooks:
                eachbook = self.mongo['book'].find_one({'id': boughtbook['book_id']},
                                                       {'_id': 0, 'id': 1, 'title': 1, 'author': 1, 'tags': 1, 'publisher':1})
                labels += eachbook['tags']
                reco[eachbook['id']] = eachbook
                reco[eachbook['id']]['tags'] = []
                books = self.mongo['book'].find({'$or': [{'author': eachbook['author']},
                                                         {'publisher': eachbook['publisher']},
                                                         {'tags': {'$elemMatch': {'$in': eachbook['tags']}}}]},
                                                {'_id': 0, 'id': 1, 'title': 1, 'author': 1, 'tags': 1})
                for book in books:
                    if book['id'] not in reco.keys():
                        reco[book['id']] = book
        print(len(reco))
        labels = list(set(labels))
        print(labels)
```

Figure: 小型推荐系统的实现

我们基于用户搜索/购买书本的历史，实现了一个小型的推荐系统

1 数据库设计

2 基本功能设计与实现

3 进阶功能设计与实现

4 接口测试与性能分析
- 功能接口测试
- 代码覆盖率
- 性能分析与提升

5 小组协作与分工

6 演示与提问

接口测试

接口测试详情

(1) 35 个基础测试全部通过

(2) 为进阶功能编写了 37 个测试

```
fe/test/test_order.py::TestSendBooks::test_processing_order_sent PASSED                 [ 62%]
fe/test/test_order.py::TestSendBooks::test_processing_order_receive PASSED              [ 63%]
fe/test/test_order.py::TestSendBooks::test_history_order PASSED                         [ 65%]
fe/test/test_order.py::TestSendBooks::test_history_order_sent PASSED                    [ 66%]
fe/test/test_order.py::TestSendBooks::test_history_order_receive PASSED                 [ 68%]
fe/test/test_order.py::TestSendBooks::test_recommend_empty PASSED                       [ 69%]
fe/test/test_order.py::TestSendBooks::test_recommend_ok PASSED                          [ 70%]
fe/test/test_order.py::TestSendBooks::test_seller_processing_order_ok PASSED            [ 72%]
fe/test/test_order.py::TestSendBooks::test_seller_processing_order_sent PASSED          [ 73%]
fe/test/test_order.py::TestSendBooks::test_seller_processing_order_receive PASSED       [ 75%]
fe/test/test_order.py::TestSendBooks::test_seller_history_order PASSED                  [ 76%]
fe/test/test_order.py::TestSendBooks::test_seller_history_order_sent PASSED             [ 77%]
fe/test/test_order.py::TestSendBooks::test_seller_history_order_receive PASSED          [ 79%]
fe/test/test_order.py::TestSendBooks::test_get_book_info_ok PASSED                      [ 80%]
fe/test/test_order.py::TestSendBooks::test_get_book_info_error PASSED                   [ 81%]
fe/test/test_order.py::TestSendBooks::test_search_in_store_ok PASSED                    [ 83%]
fe/test/test_order.py::TestSendBooks::test_search_in_store_page_ok PASSED               [ 84%]
fe/test/test_password.py::TestPassword::test_ok PASSED                                  [ 86%]
fe/test/test_password.py::TestPassword::test_error_password PASSED                      [ 87%]
fe/test/test_password.py::TestPassword::test_error_user_id PASSED                       [ 88%]
fe/test/test_payment.py::TestPayment::test_ok PASSED                                    [ 90%]
fe/test/test_payment.py::TestPayment::test_authorization_error PASSED                   [ 91%]
fe/test/test_payment.py::TestPayment::test_not_suff_funds PASSED                        [ 93%]
fe/test/test_payment.py::TestPayment::test_repeat_pay PASSED                            [ 94%]
fe/test/test_register.py::TestRegister::test_register_ok PASSED                         [ 95%]
fe/test/test_register.py::TestRegister::test_unregister_ok PASSED                       [ 97%]
fe/test/test_register.py::TestRegister::test_unregister_error_authorization PASSED      [ 98%]
fe/test/test_register.py::TestRegister::test_register_error_exist_user_id PASSED        [100%]

============================================================================== 72 passed in 186.43s (0:03:06) ==============================================================================
```

接口测试

接口测试详情

(1) 35 个基础测试全部通过

(2) 为进阶功能编写了 37 个测试

```
fe/test/test_order.py::TestSendBooks::test_processing_order_sent PASSED                  [ 62%]
fe/test/test_order.py::TestSendBooks::test_processing_order_receive PASSED               [ 63%]
fe/test/test_order.py::TestSendBooks::test_history_order PASSED                          [ 65%]
fe/test/test_order.py::TestSendBooks::test_history_order_sent PASSED                     [ 66%]
fe/test/test_order.py::TestSendBooks::test_history_order_receive PASSED                  [ 68%]
fe/test/test_order.py::TestSendBooks::test_recommend_empty PASSED                        [ 69%]
fe/test/test_order.py::TestSendBooks::test_recommend_ok PASSED                           [ 70%]
fe/test/test_order.py::TestSendBooks::test_seller_processing_order_ok PASSED             [ 72%]
fe/test/test_order.py::TestSendBooks::test_seller_processing_order_sent PASSED           [ 73%]
fe/test/test_order.py::TestSendBooks::test_seller_processing_order_receive PASSED        [ 75%]
fe/test/test_order.py::TestSendBooks::test_seller_history_order PASSED                   [ 76%]
fe/test/test_order.py::TestSendBooks::test_seller_history_order_sent PASSED              [ 77%]
fe/test/test_order.py::TestSendBooks::test_seller_history_order_receive PASSED           [ 79%]
fe/test/test_order.py::TestSendBooks::test_get_book_info_ok PASSED                       [ 80%]
fe/test/test_order.py::TestSendBooks::test_get_book_info_error PASSED                    [ 81%]
fe/test/test_order.py::TestSendBooks::test_search_in_store_ok PASSED                     [ 83%]
fe/test/test_order.py::TestSendBooks::test_search_in_store_page_ok PASSED                [ 84%]
fe/test/test_password.py::TestPassword::test_ok PASSED                                   [ 86%]
fe/test/test_password.py::TestPassword::test_error_password PASSED                       [ 87%]
fe/test/test_password.py::TestPassword::test_error_user_id PASSED                        [ 88%]
fe/test/test_payment.py::TestPayment::test_ok PASSED                                     [ 90%]
fe/test/test_payment.py::TestPayment::test_authorization_error PASSED                    [ 91%]
fe/test/test_payment.py::TestPayment::test_not_suff_funds PASSED                         [ 93%]
fe/test/test_payment.py::TestPayment::test_repeat_pay PASSED                             [ 94%]
fe/test/test_register.py::TestRegister::test_register_ok PASSED                          [ 95%]
fe/test/test_register.py::TestRegister::test_unregister_ok PASSED                        [ 97%]
fe/test/test_register.py::TestRegister::test_unregister_error_authorization PASSED       [ 98%]
fe/test/test_register.py::TestRegister::test_register_error_exist_user_id PASSED         [100%]

============================================================================== 72 passed in 186.43s (0:03:06) ==============================================================================
```

覆盖率测试

```
C:\Users\汤琼\Desktop\大三上课程\数据库\BookStore\panzer>coverage report
Name                                    Stmts   Miss Branch BrPart  Cover
-------------------------------------------------------------------------
be\__init__.py                              0      0      0      0   100%
be\app.py                                   4      4      2      0     0%
be\model\__init__.py                        0      0      0      0   100%
be\model\buyer.py                         227     52     92     18    73%
be\model\db_conn.py                        29      0      8      0   100%
be\model\error.py                          25      2      0      0    92%
be\model\nlp.py                            75      3     32      3    94%
be\model\ocr.py                             0      0      0      0   100%
be\model\order.py                          34      8     10      2    73%
be\model\seller.py                        153     29     72     11    76%
be\model\store.py                          41      3      0      0    93%
be\model\times.py                          24      4      0      0    83%
be\model\user.py                          200     39     72     10    77%
be\serve.py                                43      6      4      2    83%
be\tasks.py                                 2      0      0      0   100%
be\view\__init__.py                         0      0      0      0   100%
be\view\auth.py                            60      0      0      0   100%
be\view\buyer.py                           76      0      2      0   100%
be\view\seller.py                          50      0      0      0   100%
fe\__init__.py                              0      0      0      0   100%
fe\access\__init__.py                       0      0      0      0   100%
fe\access\auth.py                          46      0      0      0   100%
fe\access\book.py                          70      1     12      2    96%
fe\access\buyer.py                         77      0      2      0   100%
fe\access\new_buyer.py                     14      0      0      0   100%
```

覆盖率测试 Cont'd

fe\access\new_seller.py	8	0	0	0	100%
fe\access\seller.py	47	0	0	0	100%
fe\bench__init__.py	0	0	0	0	100%
fe\bench\run.py	13	0	6	0	100%
fe\bench\session.py	47	0	12	1	98%
fe\bench\workload.py	125	1	22	2	98%
fe\conf.py	11	0	0	0	100%
fe\conftest.py	17	0	0	0	100%
fe\test\gen_book_data.py	50	0	16	0	100%
fe\test\test_add_book.py	36	0	10	0	100%
fe\test\test_add_funds.py	23	0	0	0	100%
fe\test\test_add_stock_level.py	39	0	10	0	100%
fe\test\test_bench.py	6	2	0	0	67%
fe\test\test_create_store.py	20	0	0	0	100%
fe\test\test_login.py	28	0	0	0	100%
fe\test\test_new_order.py	40	0	0	0	100%
fe\test\test_order.py	240	0	0	0	100%
fe\test\test_password.py	33	0	0	0	100%
fe\test\test_payment.py	60	1	4	1	97%
fe\test\test_register.py	31	0	0	0	100%
TOTAL	2124	155	388	52	90%

Figure: 覆盖率:90%

性能分析

我们对实现的各功能接口做了吞吐量测试

性能

峰值吞吐量：44798

```
INFO:root:TPS_C=43700, NO=OK:473253 Thread_num:977 TOTAL:473253 LATENCY:0.01389596082661572 , P=OK:471499 Thread_num:976 TOTAL:472276 LATENCY:0.008469420281381713
INFO:root:TPS_C=43748, NO=OK:474231 Thread_num:978 TOTAL:474231 LATENCY:0.013895182758260842 , P=OK:472460 Thread_num:977 TOTAL:473253 LATENCY:0.008468660857187865
INFO:root:TPS_C=43796, NO=OK:475210 Thread_num:979 TOTAL:475210 LATENCY:0.013894401810561107 , P=OK:473422 Thread_num:978 TOTAL:474231 LATENCY:0.008467903566674288
INFO:root:TPS_C=43844, NO=OK:476190 Thread_num:980 TOTAL:476190 LATENCY:0.013893613800988183 , P=OK:474385 Thread_num:979 TOTAL:475210 LATENCY:0.008467148408973857
INFO:root:TPS_C=43891, NO=OK:477171 Thread_num:981 TOTAL:477171 LATENCY:0.013892829252173599 , P=OK:475349 Thread_num:980 TOTAL:476190 LATENCY:0.008466399562371623
INFO:root:TPS_C=43939, NO=OK:478153 Thread_num:982 TOTAL:478153 LATENCY:0.013892050246946182 , P=OK:476314 Thread_num:981 TOTAL:477171 LATENCY:0.008465654914645848
INFO:root:TPS_C=43987, NO=OK:479136 Thread_num:983 TOTAL:479136 LATENCY:0.013891274667666749 , P=OK:477280 Thread_num:982 TOTAL:478153 LATENCY:0.008464914431161902
INFO:root:TPS_C=44034, NO=OK:480120 Thread_num:984 TOTAL:480120 LATENCY:0.013890502558224879 , P=OK:478247 Thread_num:983 TOTAL:479136 LATENCY:0.008464178108368067
INFO:root:TPS_C=44082, NO=OK:481105 Thread_num:985 TOTAL:481105 LATENCY:0.013889727679340725 , P=OK:479215 Thread_num:984 TOTAL:480120 LATENCY:0.00846344584337346
INFO:root:TPS_C=44130, NO=OK:482091 Thread_num:986 TOTAL:482091 LATENCY:0.013888945934191103 , P=OK:480184 Thread_num:985 TOTAL:481105 LATENCY:0.008462715534474995
INFO:root:TPS_C=44178, NO=OK:483078 Thread_num:987 TOTAL:483078 LATENCY:0.013888171728249596 , P=OK:481153 Thread_num:986 TOTAL:482091 LATENCY:0.008461983181717219
INFO:root:TPS_C=44225, NO=OK:484066 Thread_num:988 TOTAL:484066 LATENCY:0.013887396861877477 , P=OK:482123 Thread_num:987 TOTAL:483078 LATENCY:0.008461250745725865
INFO:root:TPS_C=44273, NO=OK:485055 Thread_num:989 TOTAL:485055 LATENCY:0.013886625443745896 , P=OK:483093 Thread_num:988 TOTAL:484066 LATENCY:0.008460518189191484
INFO:root:TPS_C=44321, NO=OK:486045 Thread_num:990 TOTAL:486045 LATENCY:0.013885845106567707 , P=OK:484064 Thread_num:989 TOTAL:485055 LATENCY:0.008459785551296936
INFO:root:TPS_C=44368, NO=OK:487036 Thread_num:991 TOTAL:487036 LATENCY:0.01388506612357576 , P=OK:485036 Thread_num:990 TOTAL:486045 LATENCY:0.00845905289739582 8
INFO:root:TPS_C=44416, NO=OK:488028 Thread_num:992 TOTAL:488028 LATENCY:0.013884286444348987 , P=OK:486008 Thread_num:991 TOTAL:487036 LATENCY:0.008458320233567294
INFO:root:TPS_C=44464, NO=OK:489021 Thread_num:993 TOTAL:489021 LATENCY:0.01388351425643495 , P=OK:486980 Thread_num:992 TOTAL:488028 LATENCY:0.00845758756193296
INFO:root:TPS_C=44512, NO=OK:490015 Thread_num:994 TOTAL:490015 LATENCY:0.013882763811531463 , P=OK:487953 Thread_num:993 TOTAL:489021 LATENCY:0.008456858966304145
INFO:root:TPS_C=44559, NO=OK:491010 Thread_num:995 TOTAL:491010 LATENCY:0.013882016648595749 , P=OK:488926 Thread_num:994 TOTAL:490015 LATENCY:0.008456130335418859
INFO:root:TPS_C=44607, NO=OK:492006 Thread_num:996 TOTAL:492006 LATENCY:0.013881274785104739 , P=OK:489899 Thread_num:995 TOTAL:491010 LATENCY:0.008455401691423482
INFO:root:TPS_C=44655, NO=OK:493003 Thread_num:997 TOTAL:493003 LATENCY:0.01388053007905019 , P=OK:490873 Thread_num:996 TOTAL:492006 LATENCY:0.008454675065373661
INFO:root:TPS_C=44702, NO=OK:494001 Thread_num:998 TOTAL:494001 LATENCY:0.01387978658403733 , P=OK:491848 Thread_num:997 TOTAL:493003 LATENCY:0.008453954521896002
INFO:root:TPS_C=44750, NO=OK:495000 Thread_num:999 TOTAL:495000 LATENCY:0.013879038252493347 , P=OK:492824 Thread_num:998 TOTAL:494001 LATENCY:0.008453233947911745
INFO:root:TPS_C=44798, NO=OK:496000 Thread_num:1000 TOTAL:496000 LATENCY:0.013878293099903292 , P=OK:493801 Thread_num:999 TOTAL:495000 LATENCY:0.008452519489057137
```

Figure: 吞吐量一览

性能提升

我们采取了 RETURNING 关键字减少了不必要的查询

性能提升

(1) 同时进行 SELECT 和 UPDATE 操作

(2) 作用于 INSERT,UPDATE,DELETE

(3) DELETE 返回操作前的数据，INSERT,UPDATE 返回操作后的数据

对应的 MongoDB 操作：FindOneAndUpdate...

云服务一览

云数据库

(1) 阿里云 RDS

(2) 阿里云 MongoDB

云服务一览

云数据库

(1) 阿里云 RDS

(2) 阿里云 MongoDB

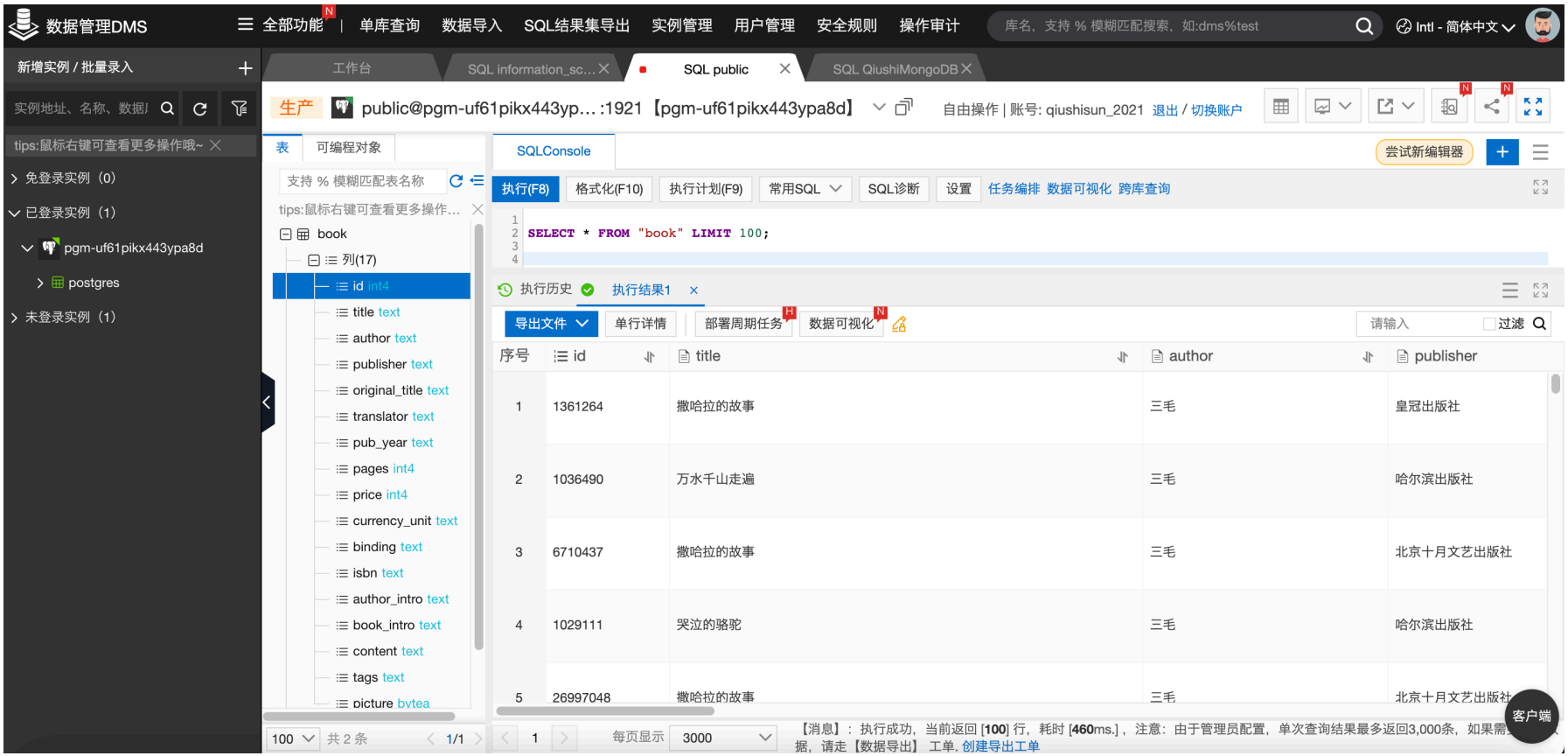


Figure: 云数据库一览

1 数据库设计

2 基本功能设计与实现

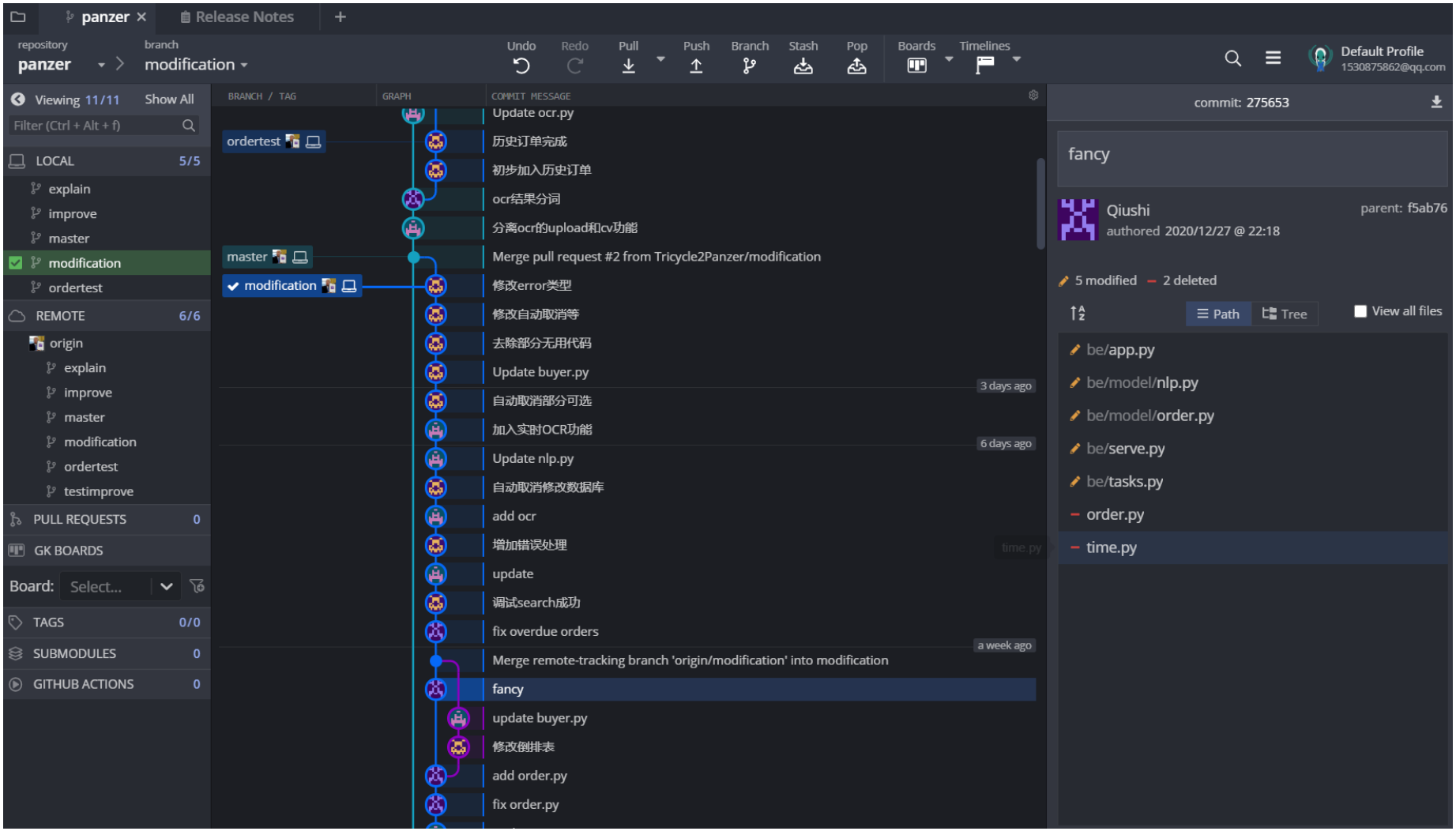
3 进阶功能设计与实现

4 接口测试与性能分析

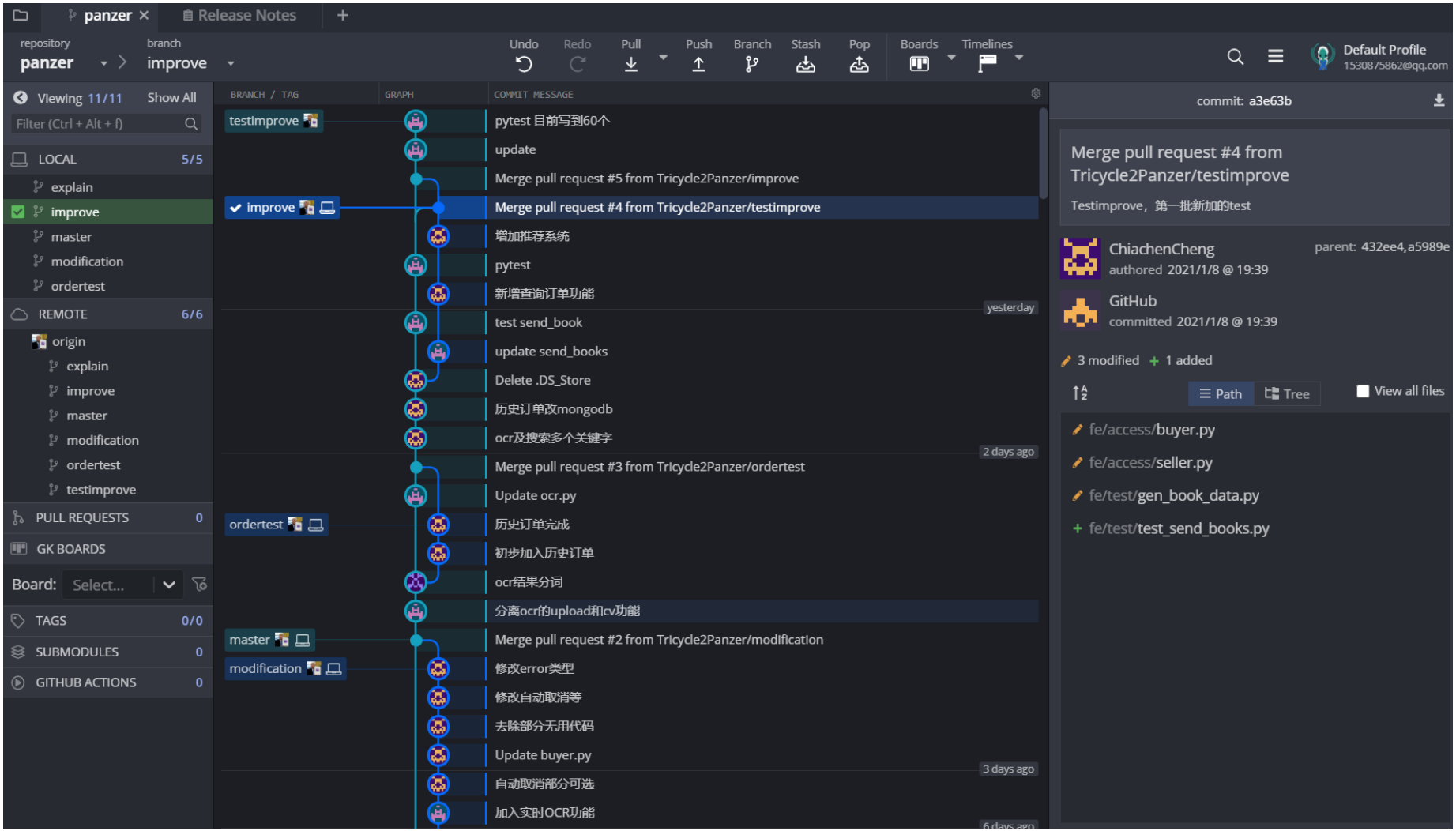
5 小组协作与分工
- Git 协作与版本控制
- 小组分工情况

6 演示与提问

Git 协作与版本控制


panzer
Release Notes
repository
panzer
branch
modification
Undo
Redo
Pull
Push
Branch
Stash
Pop
Boards
Timelines
Default Profile
1530875862@qq.com
Viewing 11/11
Show All
Filter (Ctrl + Alt + f)
LOCAL 5/5
explain
improve
master
modification
ordertest
REMOTE 6/6
origin
explain
improve
master
modification
ordertest
testimprove
PULL REQUESTS 0
GK BOARDS
Board: Select...
TAGS 0/0
SUBMODULES 0
GITHUB ACTIONS 0
BRANCH / TAG
GRAPH
COMMIT MESSAGE
Update ocr.py
ordertest
历史订单完成
初步加入历史订单
ocr结果分词
分离ocr的upload和cv功能
master
Merge pull request #2 from Tricycle2Panzer/modification
modification
修改error类型
修改自动取消等
去除部分无用代码
Update buyer.py
3 days ago
自动取消部分可选
加入实时OCR功能
6 days ago
Update nlp.py
自动取消修改数据库
add ocr
增加错误处理
update
调试search成功
fix overdue orders
a week ago
Merge remote-tracking branch 'origin/modification' into modification
fancy
update buyer.py
修改倒排表
add order.py
fix order.py
commit: 275653
fancy
Qiushi
authored 2020/12/27 @ 22:18
parent: f5ab76
5 modified
2 deleted
Path
Tree
View all files
be/app.py
be/model/nlp.py
be/model/order.py
be/serve.py
be/tasks.py
order.py
time.py

Git 协作与版本控制


panzer
Release Notes
repository
panzer
branch
improve
Undo
Redo
Pull
Push
Branch
Stash
Pop
Boards
Timelines
Default Profile
1530875862@qq.com
Viewing 11/11
Show All
Filter (Ctrl + Alt + f)
LOCAL 5/5
explain
improve
master
modification
ordertest
REMOTE 6/6
origin
explain
improve
master
modification
ordertest
testimprove
PULL REQUESTS 0
GK BOARDS
Board: Select...
TAGS 0/0
SUBMODULES 0
GITHUB ACTIONS 0
BRANCH / TAG
GRAPH
COMMIT MESSAGE
testimprove
pytest 目前写到60个
update
Merge pull request #5 from Tricycle2Panzer/improve
improve
Merge pull request #4 from Tricycle2Panzer/testimprove
增加推荐系统
pytest
新增查询订单功能
yesterday
test send_book
update send_books
Delete .DS_Store
历史订单改mongodb
ocr及搜索多个关键字
2 days ago
Merge pull request #3 from Tricycle2Panzer/ordertest
Update ocr.py
ordertest
历史订单完成
初步加入历史订单
ocr结果分词
分离ocr的upload和cv功能
master
Merge pull request #2 from Tricycle2Panzer/modification
modification
修改error类型
修改自动取消等
去除部分无用代码
Update buyer.py
3 days ago
自动取消部分可选
加入实时OCR功能
6 days ago
commit: a3e63b
Merge pull request #4 from Tricycle2Panzer/testimprove
Testimprove，第一批新加的test
ChiachenCheng
authored 2021/1/8 @ 19:39
parent: 432ee4,a5989e
GitHub
committed 2021/1/8 @ 19:39
3 modified + 1 added
Path
Tree
View all files
fe/access/buyer.py
fe/access/seller.py
fe/test/gen_book_data.py
fe/test/test_send_books.py

小组分工情况

分工明细

(1) **郑佳辰**：数据库设计，推荐系统，倒排表及搜索系统，历史订单，性能优化，接口测试，产品经理

(2) **孙秋实**：订单自动取消，基本功能实现，接口测试，吞吐量测试，项目上云，slides 制作

(3) **汤琼**: 接口测试，基本功能实现，以图搜书，实时 OCR，postman 测试，收发货功能实现，吞吐量测试

Git 版本控制和项目文档维护由三位同学一起完成。

小组分工情况

分工明细

(1) **郑佳辰**：数据库设计，推荐系统，倒排表及搜索系统，历史订单，性能优化，接口测试，产品经理

(2) **孙秋实**：订单自动取消，基本功能实现，接口测试，吞吐量测试，项目上云，slides 制作

(3) **汤琼**: 接口测试，基本功能实现，以图搜书，实时 OCR，postman 测试，收发货功能实现，吞吐量测试

Git 版本控制和项目文档维护由三位同学一起完成。

小组分工情况

分工明细

(1) **郑佳辰**：数据库设计，推荐系统，倒排表及搜索系统，历史订单，性能优化，接口测试，产品经理

(2) **孙秋实**：订单自动取消，基本功能实现，接口测试，吞吐量测试，项目上云，slides 制作

(3) **汤琼:** 接口测试，基本功能实现，以图搜书，实时 OCR，postman 测试，收发货功能实现，吞吐量测试

Git 版本控制和项目文档维护由三位同学一起完成。

1 数据库设计

2 基本功能设计与实现

3 进阶功能设计与实现

4 接口测试与性能分析

5 小组协作与分工

6 演示与提问

演示与提问

Demo and Questions!