

**数据库系统大作业设计文档**

**实验题目：网上银行储蓄业务系统**

**学 院：统计学院**

**专 业：数据科学与大数据技术**

**年 级：本科二年级**

**学 号：2018201670**

**姓 名：白迎辰**

**指导教师：覃雄派**

**完成日期：2020年6月1日**

**1 前言**

**1.1** 网上银行又称网络银行、在线银行或电子银行，它是各银行在互联网中设立的虚拟柜台，银行利用网络技术，通过互联网向客户提供开户、销户、查询、对账、行内转账、跨行转账、信贷、网上证券、投资理财等传统服务项目，使客户足不出户就能够安全、便捷地管理活期和定期存款、支票、信用卡及个人投资等。

**1.2** 此项目基于Python、HTML、SQL三种语言，通过pyodbc连接数据库，flask架构网站，实现了具备增、删、查、该四种基本操作功能，开户、销户、绑定/解绑储蓄卡、存钱、取钱、转账、计息、查看余额与交易历史等基本业务功能以及数据可视化功能；支持用户传入参数（用户名，账号）通过网页互动在同一页面上动态生成特定查询结果及其汇总统计图表；支持一次操作为账户绑定两张储蓄卡；支持操作结果及时反馈：会在按钮附近向用户反馈；引入权限列表，防止非法用户通过网址跳过登录环节直接访问网站

**2 业务描述**

**（一） 基本功能要求**

1. 一家银行有很多的储户（用户），每个用户可以开设多个账户，每个储户针对各个账户可以开两个储蓄卡主卡和副卡，比如主卡自己用，副卡家人用。

储户的信息包括身份证号、姓名、性别、出生年月、电话、邮箱、邮编、地址等信息。

账户的信息包括账号、销户标记、账户余额等信息。

储蓄卡的信息包括主卡/还是副卡等信息。

2. 用户开户时，同时建立主卡/副卡。储户可以用主卡或者副卡，对银行账户进行销户、存入、取出、转账等操作，需要记录一个交易历史记录。

每条交易历史包括账户、主卡还是副卡、存入/取出/还是转账、对方账号、摘要、金额、交易之前的余额、交易后的余额等信息。

一个账户对应很多的交易记录，记录了开户以来进行的历史交易。开户也需要登记一个历史记录。

需要注意的是，一般存入金额都是可行的。但是取钱，前提条件必须是账户里面有足够的钱，转账则需要转出的账户有足够的钱（余额）。

3. 银行每年固定时间即某天，给银行账户计息。

计息规则是，根据上次计息以来，该账户在每个时间段的账户余额，进行计息，然后加起来，得到利息总额。一般是半夜计息。

比如，上次计息日为2020-1-1，之后到2020-1-2账户余额一直是10000元，后来有存入，到2020-1-5账户余额一直是20000元。2020-1-5计息。

那么，按照10000元持续1天，20000元持续3天计息，假设年利率为1.5%，那么两部分利息为，1\*10000\*1.5%/365，3 \*2000×1.5%/365，加起来作为这段时间的账户利息。

需要记录计息历史，包括账户、上次计息日、本次计息日、计息金额。

并且在交易历史里，记录计息操作，即存入计算出来的利息，摘要部分记录为“计息”。

**（二） 基本功能设计思路**

1. 开户：

在用户表中以元组的形式插入账户持有者的基本信息

在账户表中插入新账户相关信息

在储蓄卡表中插入新账户对应储蓄卡的信息

在交易历史记录中插入信息（摘要处为"开户"）

2. 销户

在用户表中删除账户持有者对应的元组

在账户表中删除账户对应元组

在储蓄卡表中删除对应储蓄卡信息

在交易历史记录中插入信息（摘要处为"销户"）

3. 存入

重新计算账户余额

在交易历史中记录存入操作，摘要部分记录为“存入”

4. 取出

判断当前账户余额是否大于等于取出金额，若否，则拒绝操作

重新计算账户余额

在交易历史中记录存入操作，摘要部分记录为“取出”

5. 转账

判断当前账户余额是否大于等于转账金额，若否，则拒绝操作

重新计算账户余额

在交易历史中记录存入操作，摘要部分记录为“转账”

6. 记录交易历史

在交易历史记录表中以元组形式插入操作的相关信息

7. 账户计息

查询账户计息表中最后一条元组中本次计息日之后对应的所有交易历史

计算出这段时间内该账户余额随时间的变动关系

计算出不同时段不同账户余额的利息和作为账户利息

记录计息历史，包括账户、上次计息日、本次计息日、计息金额

在交易历史记录计息操作，存入计算利息，摘要部分记为“利息”

**（三） 完整功能实现**

1. 登录

若用户点击“登录”按钮：

判断用户ID与密码是否输入完整，若不完整则停留在页面

若完整，根据用户ID与密码查询Userinfo表

若查询结果存在，将ID放置在权限列表permission中

页面跳转到/<ID>/Index.html，ID作为一级参数

若用户点击“注册”按钮：

页面跳转到/Register.html

2. 注册

若用户点击“确认”：

检查基本信息是否填写完整

若不完整则停留在原页面

检查用户名是否已经被占用

若占用则停留在原页面

检查用户在注册界面输入的两次密码是否相同

若相同则停留在原页面

向Userinfo表中插入注册信息

页面跳转到登录界面/Login.html

3. 主菜单

检查用户是否已经登录

若用户点击“账户信息”

跳转到新页面动态生成用户基本信息

若用户点击“管理账户”

跳转到账户菜单界面/<ID>/my\_account.html

若用户点击“返回”，则跳转到登录界面

4. 账户菜单

检查用户是否已经登录

若用户点击“新建账户”

跳转到开户界面/<ID>/new\_account.html

若用户点击“登入账户”

跳转到登录账户界面/<ID>/login\_account.html

若用户点击“注销账户”

跳转到账户注销界面/<ID>/delete\_account.html

若用户点击“返回”

跳转到主菜单/<ID>/Index.html

5. 开户

检查用户是否已经登录

若用户点击“提交”

检查用户是否输入账户

若不完整则停留在该页面

检查账户是否已经存在

若存在则停留在页面

向Account表中插入新建账户

页面跳转到/<ID>/my\_account.html

若用户点击“返回”

返回/<ID>/my\_account.html

6.销户

检查用户是否已经登录

若用户点击“提交”

检查用户是否输入账户

更新Account中对应账户的销户标记

将账户绑定的所有储蓄卡从Account\_Card表中删除

向Trade\_History表中插入用户销户记录

页面跳转到/<ID>/my\_account.html

若用户点击“返回”

页面跳转到/<ID>/my\_account.html

7.登录账户

检查用户是否登录

若用户点击“提交”

检查用户是否输入账户

检查账户是否存在

若不存在则返回/<ID>/my\_account.html

若存在则跳转到账户中心/<ID>/<account>/account\_center.html

若用户点击“返回”

页面跳转到/<ID>/my\_account.html

8.账户中心

检查用户是否登录

若用户点击“查看账户余额信息”

查询账户相关所有信息

若账户不存在则返回/<ID>/<account>/account\_center.html

动态生成用户绑定储蓄卡的余额信息

绘制储蓄卡余额柱状图，绑定几张绘制几张

若用户点击“绑定”

页面跳转到/<ID>/<account>/Bind.html

若用户点击“存钱”

页面跳转到/<ID>/<account>/Deposit.html

若用户点击“取钱”

页面跳转到/<ID>/<account>/Withdraw.html

若用户点击“转账”

页面跳转到/<ID>/<account>/Transfer.html

若用户点击“查看账户交易历史”

提取所有该用户在该账号上的历史操作记录

动态生成账户交易历史

绘制储蓄卡余额变化的折线图

若用户点击“查看账户利息”

页面跳转到/<ID>/<account>/Interest.html

若用户点击“返回”

页面跳转到/<ID>/login\_account.html

9.绑定/解绑银行卡

检查用户是否登录

若用户点击“绑定”

检查用户是否填写卡号与卡种信息

获得用户填写的卡号与卡种

检查该卡是否已被绑定

检查该用户在该账户上是否绑定了超过两张卡

向Account\_Card表中插入绑定卡的信息

向Trade\_History表中插入用户绑定储蓄卡的历史记录

若用户点击“解绑”

检查用户是否填写卡号与卡种信息

获得用户填写的卡号与卡种

检查该卡是否已被绑定

从Account\_Card表中删去此卡的信息

向Trade\_History表中插入用户解除绑定储蓄卡的历史记录

若用户点击“返回”

页面跳转到/<ID>/<account>/account\_center.html

10.存钱

检查用户是否登录

若用户点击“提交”

检查用户是否填写卡号与金额

获得存款卡号与金额

检查该卡是否与账户绑定

更新卡的余额

向Trade\_History表中插入用户向该卡存钱的历史记录

重新计算账户总余额

若该账户开通后首次存钱

向表Cal\_Interest中插入一条初始计息历史

若该账户不是开通后首次存钱

调用计息函数计算上次存取到本次存取所获得的利息

若用户点击“返回”

页面跳转到/<ID>/<account>/account\_center.html

11.取钱

检查用户是否登录

若用户点击“提交”

检查用户是否填写卡号与金额

获得取钱卡号与金额

检查该卡是否与账户绑定

更新卡的余额

向Trade\_History表中插入用户向该卡存钱的历史记录

重新计算账户总余额

若该账户开通后首次存钱

向表Cal\_Interest中插入一条初始计息历史

若该账户不是开通后首次存钱

调用计息函数计算上次存取到本次存取所获得的利息

若用户点击“返回”

页面跳转到/<ID>/<account>/account\_center.html

12.转账

检查用户是否登录

若用户点击“提交”

检查用户是否输入本人卡号，对方卡号与转账金额

获取本人卡号，对方卡号与转账金额

检查本人卡号是否绑定账户

检查对方卡号是否绑定账户

检查本人卡上余额是否足够

更新本人卡上余额

更新本人账户余额

向Trade\_History表中插入本人发起转账的历史记录

调用计息函数

更新对方卡上余额

更新对方账户余额

向Trade\_History表中插入对方接受转账的历史记录

调用计息函数

若用户点击“返回”

页面跳转到/<ID>/<account>/account\_center.html

13.账户利息管理

检查用户是否登录

若用户点击“查看历史计息”

从Cal\_Inerest表中提取账户所有历史计息记录

动态生成历史计息记录

若用户点击“获得收益”

检查该账户是否存在初始计息记录

向计息记录中添加一个计息记录，由于用户提取了利息，过去的累积利息归零

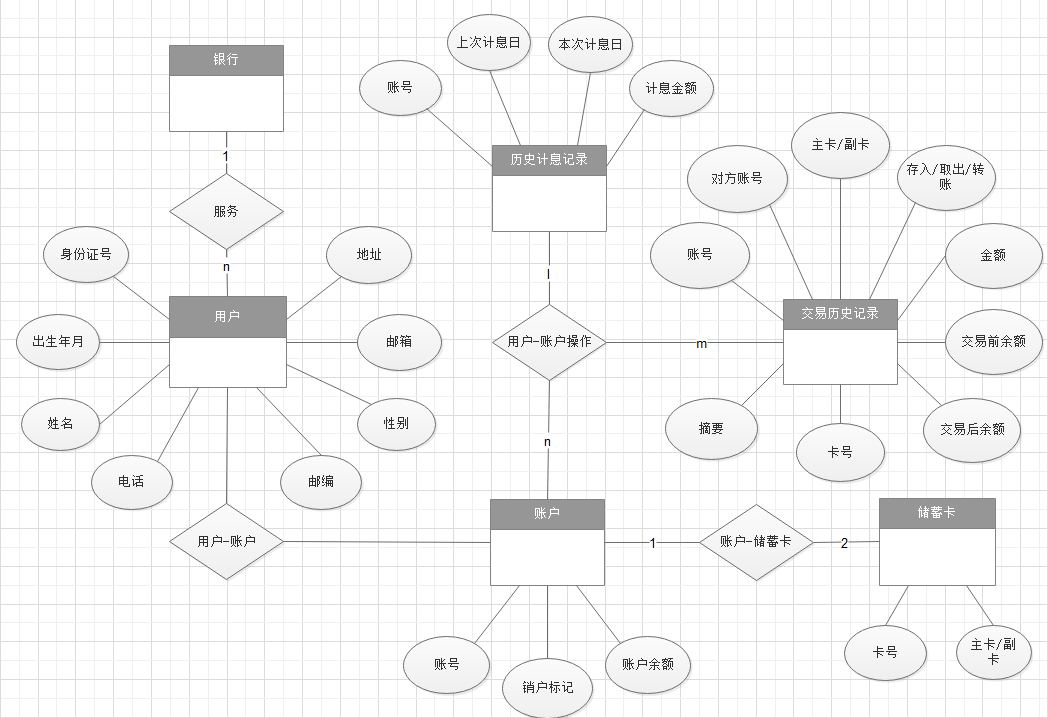
将利息转入用户的储蓄卡中（这里选择了距今最短时间内活跃的储蓄卡）

向Trade\_History表中插入储蓄卡获得利息的记录

更新账户余额

**3 数据库设计**

**（一） ER图设计**



**（二） 关系模式设计**

用户（用户ID，出生年月，姓名，电话，邮箱，性别，邮编，家庭住址，用户密码）

账户（账号，销户记录，账户余额，用户ID）

储蓄卡（卡号，账号，主卡/副卡，储蓄卡余额）

历史计息记录 （账号，上次计息日， 本次计息日， 计息金额）

交易历史记录 （用户ID，摘要，本人账号，对方账号， 主卡/副卡，存入/取出/转账/利息/绑定储蓄卡/销户，金额，交易前余额，交易后余额，卡号，操作时间）

**4 系统设计**

**（一） 应用系统模块设计**

**（二） 功能点描述**

**（三） 系统实现描述**

**3.1物理设计**

实现系统设计环境为Windows10 + MS SQL Server 2019 + PYODBC + Flask框架

3.1.1 建立用户信息表

create table Userinfo

(

ID char(18) primary key,

birth datetime not null,

username char(50) not null,

telephone char(20) not null,

email char(50) not null,

gender char(10) not null check( 性别 in(' 男',' 女')),

postcode char(10) not null,

address char(50) not null,

password char(50) not null

)

3.1.2 建立账户信息表

create table Account

(

account char(12) primary key,

balance decimal(20,2),

logoff char(10) not null,

ID char(18) not null

foreign key(ID) references Userinfo(ID)

)

3.1.3 建立储蓄卡信息表

create table Account\_Card

(

card\_num char(12) primary key

account char(12) not null,

card\_type char(10) not null,

balance decimal(20,2) not null

foreign key(account) references Account(account)

)

3.1.4 建立交易历史信息表

create table Account\_Card

(

ID char(18)

summary ntext not null,

my\_account char(12) not null,

sb\_account char(12) not null,

card\_type char(10) not null,

operation char(10) not null,

money decimal(18, 2) not null,

pre\_balance decimal(18, 2) not null,

post\_balance decimal(18, 2) not null,

card\_num char(12) not null,

OrderDate datetime not null

foreign key(account) references Account,Account\_Card(account)

)

3.1.4 建立计息表

create table Cal\_Interest

(

account char(12) not null,

pre\_calc date

cur\_calc date

interest decimal(18, 2)

foreign key(account) references Account,Account\_Card(account)

)

**3.2 数据库连接实现**

from flask import Flask,url\_for,request,render\_template,redirect,flash,render\_template\_string,Response

from datetime import datetime

import pyodbc

# PC服务器参数设置

default\_server = 'localhost'

default\_database = 'bank'

default\_username = 'sa'

default\_password = 'qwaszx'

# 权限字典

permission = {}

class PYSQL:

    def \_\_init\_\_(self, server, database, username, password):

        self.server = server

        self.database = database

        self.username = username

        self.password = password

    def GetConnect(self):

        # 连接数据库返回游标

        if not self.database:

            raise NameError("database not found")

        self.conn = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server};SERVER=' + self.server + ';DATABASE=' + self.database + ';UID='+ self.username + ';PWD=' + self.password + ';CHARSET=GBK')

        # 获得游标

        cur = self.conn.cursor()

        if not cur:

            raise NameError("fail to connect database")

        else:

            return cur

    def ExecQuery(self,sql):

        # 执行sql查询语句并返回查询结果

        cur = self.GetConnect()

        cur.execute(sql)

        # 返回查询结果

        resList = cur.fetchall()

        self.conn.close()

        return resList

    def ExecNonQuery(self,sql):

        # 执行查询语句不返回查询结果

        cur = self.GetConnect()

        cur.execute(sql)

        self.conn.commit()

        self.conn.close()

# 初始化app对象

app = Flask(\_\_name\_\_)

def init():

    global DB

    DB = PYSQL(default\_server, default\_database, default\_username, default\_password)

if \_\_name\_\_ == "\_\_main\_\_":

    init()

    app.run(debug=True, port=5000)

**3.3 Web架构**

3.2.1 Login.html



# 用户登录

@app.route('/',methods=['POST','GET'])

def login():

    if request.method == 'GET':

        return render\_template('Login.html',message="")

    if request.method =='POST':

        if 'Login' in request.form.keys():

            ID=request.form['ID']

            password=request.form['password']

            if ID=='' or password=='':

                message=""

            else:

                A='''

                select \* from Userinfo

                where ID='{ID}' and password='{password}'

                '''.format(ID=ID,password=password)

                result=DB.ExecQuery(A)

                if len(result)!=0:

                    A='''

                    select account from Userinfo

                    where ID='{ID}' and password='{password}'

                    '''.format(ID=ID,password=password)

                    permission[str(ID)]=1

                    return redirect(url\_for('Index',ID=str(ID)))

                A='''

                select \* from Userinfo

                where ID='{ID}'

                '''.format(ID=ID)

                result=DB.ExecQuery(A)

                if len(result)==0:

                    message="错误:用户名不存在"

                else:

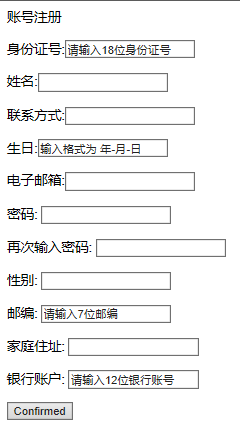
                    message="错误:密码错误"

            return render\_template('Login.html',message=message)

        if 'Register' in request.form.keys():

            return redirect(url\_for('Register'))

3.2.2 Register.html



# 用户注册

@app.route('/Register', methods=['POST', 'GET'])

def Register():

    if request.method =='GET':

        return render\_template('Register.html')

    if request.method =='POST':

        if 'confirmed' in request.form.keys():

            message=""

            ID = request.form['ID']

            username = request.form['username']

            telephone = request.form['telephone']

            birth = request.form['birth']

            email = request.form['email']

            password1 = request.form['password1']

            password2 = request.form['password2']

            gender = request.form['gender']

            postcode = request.form['postcode']

            address = request.form['address']

            account = request.form['account']

            if ID=="" or username=="" or birth=="" or email=="" or password1=="" or password2=="" or gender=="" or postcode=="" or address=="" or account=="":

                message="错误: 用户信息不完整"

                return render\_template('Register.html',message=message)

            A='''

            select \* from Userinfo

            where ID='{ID}'

            '''.format(ID = ID)

            result=DB.ExecQuery(A)

            if len(result)!=0:

                message='错误: 用户已存在'

            if password1!=password2:

                message='错误: 两次输入密码不一致'

            if message=="":

                A='''

                insert

                into

                Userinfo

                values('{uid}','{ubirth}','{uname}','{utele}','{uemail}','{ugender}','{upostcode}','{uaddress}','{upwd}')

                '''.format(uid=ID,ubirth = birth,uname=username,utele=telephone,uemail=email,ugender=gender,upostcode=postcode,uaddress=address,upwd=password1)

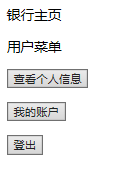
                DB.ExecNonQuery(A)

                return redirect(url\_for('login'))

            else:

                return render\_template('Register.html',message=message)

3.2.3 Index.html



# 主菜单

@app.route('/<ID>/Index',methods=['POST', 'GET'])

def Index(ID):

    if ID not in permission.keys():

        permission[ID]=0

    if permission[ID]==0:

        return "请先登录再进行此操作"

    if request.method =='GET':

        return render\_template('Index.html')

    if request.method =='POST':

        if 'Info' in request.form.keys():

            message=""

            A='''

            select ID,birth,username,telephone,email,gender,postcode,address,password

            from Userinfo

            where ID='{ID}'

            '''.format(ID=ID)

            if message=="":

                cursor = DB.GetConnect()

                s = "<table style='border:1px solid red'>"

                s = s + "<tr><td>用户ID</td><td>生日</td><td>用户名</td><td>电话</td><td>邮箱</td><td>性别</td><td>邮编</td><td>居住地址</td><td>当前用户密码</td></tr>"

                with cursor.execute(A):

                    row = cursor.fetchone()

                    while row:

                        s = s + "<tr>"

                        s = s + "<td>" + str(row[0]) + "</td>"

                        print(str(row[0]))

                        s = s + "<td>" + str(row[1]) + "</td>"

                        print(str(row[1]))

                        s = s + "<td>" + row[2].encode('gbk').decode('gbk') + "</td>"

                        print(row[2].encode('gbk').decode('gbk'))

                        s = s + "<td>" + str(row[3]) + "</td>"

                        print(str(row[3]))

                        s = s + "<td>" + str(row[4]) + "</td>"

                        print(str(row[4]))

                        s = s + "<td>" + str(row[5]) + "</td>"

                        print(str(row[5]))

                        s = s + "<td>" + str(row[6]) + "</td>"

                        print(str(row[6]))

                        s = s + "<td>" + str(row[7]) + "</td>"

                        print(str(row[7]))

                        s = s + "<td>" + str(row[8]) + "</td>"

                        print(str(row[8]))

                        row = cursor.fetchone()

                        s = s +  "</tr>"

                s = s + "</table>"

                s = "<html><body>" + s +"</body></html>"

                return render\_template\_string(s)

        if 'MA' in request.form.keys():

            return redirect(url\_for('my\_account',ID=ID))

        if 'exit' in request.form.keys():

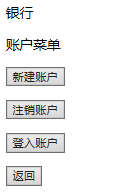
            permission[ID]=0

            return redirect(url\_for('login'))

3.2.4 查看个人信息



3.2.5 my\_account.html



# 账户菜单

@app.route('/<ID>/my\_account',methods=['POST', 'GET'])

def my\_account(ID):

    if ID not in permission.keys():

        permission[ID]=0

    if permission[ID]==0:

        return "请先登录再进行此操作"

    if request.method =='GET':

        return render\_template('my\_account.html')

    if request.method =='POST':

        if 'new\_account' in request.form.keys():

            return redirect(url\_for('new\_account',ID=ID))

        if 'login\_account' in request.form.keys():

            return redirect(url\_for('login\_account',ID=ID))

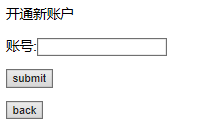
        if 'delete\_account' in request.form.keys():

            return redirect(url\_for('delete\_account',ID=ID))

        if 'back' in request.form.keys():

            return redirect(url\_for('Index',ID=ID))

3.2.6 new\_account.html



# 开户

@app.route('/<ID>/new\_account', methods=['POST', 'GET'])

def new\_account(ID):

    if ID not in permission.keys():

        permission[ID]=0

    if permission[ID]==0:

        return "请先登录再进行此操作"

    if request.method =='GET':

        return render\_template('new\_account.html')

    if request.method =='POST':

        if 'submit' in request.form.keys():

            message=""

            account=request.form['account']

            if account=="":

                message="错误：请输入账号"

                return render\_template('new\_account.html',message=message)

            A='''

            select \* from Account

            where account='{account}'

            '''.format(ID=ID,account=account)

            result=DB.ExecQuery(A)

            if len(result)!=0:

                message='错误：账号已经存在'

                return render\_template('new\_account.html',message=message)

            if message=="":

                A='''

                insert

                into

                Account

                values('{account}','0','False','{ID}')

                '''.format(account=account,ID=ID)

                DB.ExecNonQuery(A)

                B='''

                insert

                into

                Trade\_History(ID,summary,my\_account,sb\_account,card\_type,operation,money,pre\_balance,post\_balance,card\_num)

                values('{ID}','用户开通账户','{account}',' ',' ','开户','0','0','0',' ')

                '''.format(ID=ID,account=account)

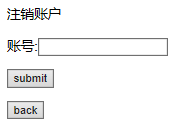
                DB.ExecNonQuery(B)

                return redirect(url\_for('my\_account',ID=ID))

        if 'back' in request.form.keys():

                return redirect(url\_for('my\_account',ID=ID))

3.2.7 delete\_account.html



# 销户

@app.route('/<ID>/delete\_account', methods=['POST', 'GET'])

def delete\_account(ID):

    if ID not in permission.keys():

        permission[ID]=0

    if permission[ID]==0:

        return "请先登录再进行此操作"

    if request.method =='GET':

        return render\_template('delete\_account.html')

    if request.method =='POST':

        if 'submit' in request.form.keys():

            message=""

            account=request.form['account']

            if account=="":

                message="错误：请输入账号"

                return render\_template('delete\_account.html',message=message)

            A='''

            select \* from Account

            where account='{account}'

            '''.format(ID=ID,account=account)

            result=DB.ExecQuery(A)

            if len(result)==0:

                message='错误：账号不存在'

                return render\_template('delete\_account.html',message=message)

            if message=="":

                A='''

                update Account

                set logoff='True'

                where account='{account}' and ID='{ID}'

                '''.format(account=account,ID=ID)

                DB.ExecNonQuery(A)

                B='''

                delete

                from

                Account\_Card

                where account='{account}'

                '''.format(account=account)

                DB.ExecNonQuery(B)

                C='''

                insert

                into

                Trade\_History(ID,summary,my\_account,sb\_account,card\_type,operation,money,pre\_balance,post\_balance,card\_num)

                values('{ID}','用户注销账户','{account}',' ',' ','销户',' ',' ',' ',' ')

                '''.format(ID=ID,account=account)

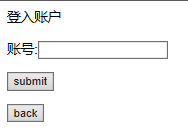
                DB.ExecNonQuery(C)

                return redirect(url\_for('my\_account',ID=ID))

        if 'back' in request.form.keys():

                return redirect(url\_for('my\_account',ID=ID))

3.2.8 Login\_account.html



@app.route('/<ID>/login\_account', methods=['POST', 'GET'])

def login\_account(ID):

    if ID not in permission.keys():

        permission[ID]=0

    if permission[ID]==0:

        return "请先登录再进行此操作"

    if request.method =='GET':

        return render\_template('login\_account.html')

    if request.method =='POST':

        if 'submit' in request.form.keys():

            message=""

            account=request.form['account']

            if account=="":

                message="错误：请输入登录账号"

                return render\_template('login\_account.html',message=message)

            A='''

            select \* from Account

            where ID='{ID}' and account='{account}' and logoff='False'

            '''.format(ID=ID,account=account)

            result=DB.ExecQuery(A)

            if len(result)==0:

                message='错误：账户不存在'

                return redirect(url\_for('my\_account',ID=ID, message=message))

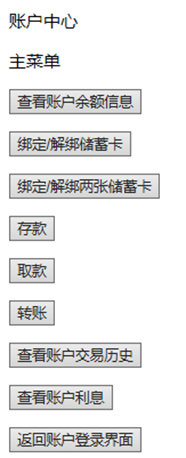
            if message=="":

                return redirect(url\_for('account\_center',ID=ID,account=account))

        if 'back' in request.form.keys():

            return redirect(url\_for('my\_account',ID=ID))

3.2.9 account\_center.html



# 账户中心

@app.route('/<ID>/<account>/account\_center',methods=['POST','GET'])

def account\_center(ID,account):

    if ID not in permission.keys():

        permission[ID]=0

    if permission[ID]==0:

        return "请先登录再进行此操作"

    if request.method =='GET':

        return render\_template('account\_center.html',ID=ID,account=account)

    if request.method =='POST':

        if 'Info' in request.form.keys():

            message=""

            A='''

            select Account.account,Account\_Card.card\_num,Account\_Card.card\_type,Account\_Card.balance

            from Account

            inner join Account\_Card

            ON Account.account = Account\_Card.account

            where Account.ID='{ID}' and Account.account='{account}' and logoff = 'False'

            '''.format(ID=ID,account=account)

            result=DB.ExecQuery(A)

            if len(result)==0:

                message='账户未开通'

                return render\_template('account\_center.html',ID=ID,account=account)

            if message=="":

                cursor = DB.GetConnect()

                s = "<table style='border:1px solid red'>"

                s = s + "<tr><td>账号</td><td>储蓄卡卡号</td><td>储蓄卡类型</td><td>储蓄卡余额</td></tr>"

                t = """<h3>账户当前余额</h3>

                <img src="/balance\_bar/{{ID}}/{{account}}.png"

                alt="balance bar as png"

                height="700"

                >"""

                with cursor.execute(A):

                    row = cursor.fetchone()

                    while row:

                        s = s + "<tr>"

                        s = s + "<td>" + str(row[0]) + "</td>"

                        print(str(row[0]))

                        s = s + "<td>" + str(row[1]) + "</td>"

                        print(str(row[1]))

                        s = s + "<td>" + row[2].encode('gbk').decode('gbk') + "</td>"

                        print(row[2].encode('gbk').decode('gbk'))

                        s = s + "<td>" + str(row[3]) + "</td>"

                        print(str(row[3]))

                        row = cursor.fetchone()

                        s = s +  "</tr>"

                s = s + "</table>"

                s = "<html><body>" + s + t +"</body></html>"

                return render\_template\_string(s,ID=ID,account=account)

        if 'Bind' in request.form.keys():

            return redirect(url\_for('Bind',ID=ID,account=account))

        # 一次性绑定/解绑两张卡

        if 'Bind2' in request.form.keys():

            return redirect(url\_for('Bind2',ID=ID,account=account))

        if 'Deposit' in request.form.keys():

            return redirect(url\_for('Deposit',ID=ID,account=account))

        if 'Withdraw' in request.form.keys():

            return redirect(url\_for('Withdraw',ID=ID,account=account))

        if 'Transfer' in request.form.keys():

            return redirect(url\_for('Transfer',ID=ID,account=account))

        if 'History' in request.form.keys():

            message=""

            A='''

            select \* from Trade\_History

            where ID ='{ID}' and my\_account='{account}'

            order by OrderDate

            '''.format(ID=ID,account=account)

            result=DB.ExecQuery(A)

            if len(result)==0:

                message='账户未开通'

                return render\_template('account\_center.html',ID=ID,account=account)

            if message=="":

                cursor = DB.GetConnect()

                s = "<table style='border:1px solid red'>"

                s = s + "<tr><td>用户</td><td>摘要</td><td>我的账号</td><td>他人的账号</td><td>储蓄卡类型</td><td>操作</td><td>金额</td><td>交易前余额</td><td>交易后余额</td><td>卡号</td><td>操作时间</td></tr>"

                t = """<h3>账户余额变化</h3>

                <img src="/balance\_variation/{{ID}}/{{account}}.png"

                alt="balance broken as png"

                height="700"

                >"""

                with cursor.execute(A):

                    row = cursor.fetchone()

                    while row:

                        s = s + "<tr>"

                        s = s + "<td>" + str(row[0]) + "</td>"

                        print(str(row[0]))

                        s = s + "<td>" + row[1].encode('gbk').decode('gbk') + "</td>"

                        print(row[1].encode('gbk').decode('gbk'))

                        s = s + "<td>" + str(row[2]) + "</td>"

                        print(str(row[2]))

                        s = s + "<td>" + str(row[3]) + "</td>"

                        print(str(row[3]))

                        s = s + "<td>" + row[4].encode('gbk').decode('gbk') + "</td>"

                        print(row[4].encode('gbk').decode('gbk'))

                        s = s + "<td>" + row[5].encode('gbk').decode('gbk') + "</td>"

                        print(row[5].encode('gbk').decode('gbk'))

                        s = s + "<td>" + str(row[6]) + "</td>"

                        print(str(row[6]))

                        s = s + "<td>" + str(row[7]) + "</td>"

                        print(str(row[7]))

                        s = s + "<td>" + str(row[8]) + "</td>"

                        print(str(row[8]))

                        s = s + "<td>" + str(row[9]) + "</td>"

                        print(str(row[9]))

                        s = s + "<td>" + (str(row[10]))[:19] + "</td>"

                        print((str(row[10]))[:19])

                        row = cursor.fetchone()

                        s = s +  "</tr>"

                s = s + "</table>"

                s = "<html><body>" + s + t +"</body></html>"

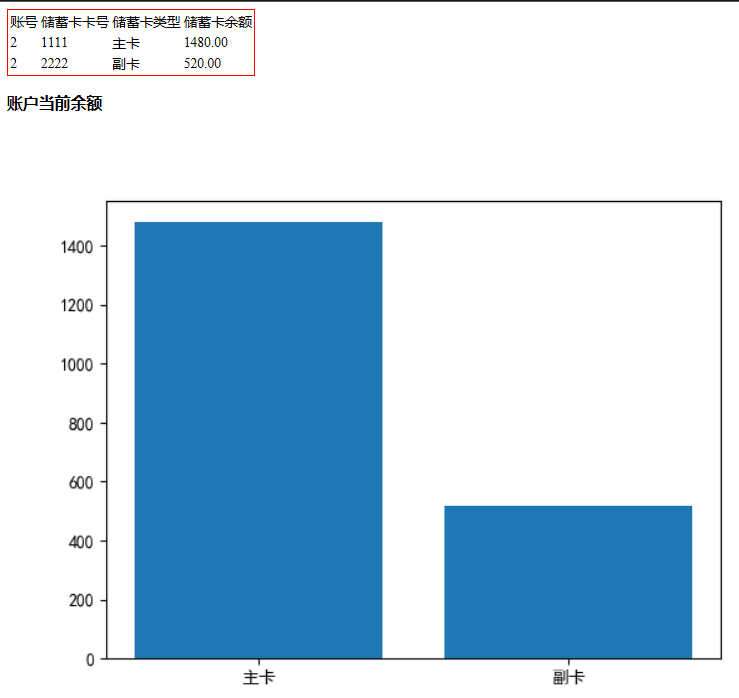
                return render\_template\_string(s,ID=ID,account=account)

        if 'Interest' in request.form.keys():

            return redirect(url\_for('Interest',ID=ID,account=account))

        if 'exit' in request.form.keys():

            return redirect(url\_for('login\_account',ID=ID))

3.2.10 查看账户余额信息

# 显示当前账户余额

@app.route("/balance\_bar/<int:ID>/<int:account>.png")

def bar\_plot\_png(ID,account):

    fig = Figure()

    axis = fig.add\_subplot(1, 1, 1)

    axis.set\_ylabel('金额')

    A='''

    select Account\_Card.balance

    from Account

    inner join Account\_Card

    ON Account.account = Account\_Card.account

    where Account.ID='{ID}' and Account.account='{account}' and logoff = 'False'

    '''.format(ID=ID,account=account)

    result=DB.ExecQuery(A)

    B='''

    select \*

    from Account

    inner join Account\_Card

    ON Account.account = Account\_Card.account

    where Account.ID='{ID}' and Account.account='{account}' and logoff = 'False' and Account\_Card.card\_type = '主卡'

    '''.format(ID=ID,account=account)

    prime=DB.ExecQuery(B)

    C='''

    select \*

    from Account

    inner join Account\_Card

    ON Account.account = Account\_Card.account

    where Account.ID='{ID}' and Account.account='{account}' and logoff = 'False' and Account\_Card.card\_type = '副卡'

    '''.format(ID=ID,account=account)

    vice=DB.ExecQuery(C)

    if len(result)==0:

        names=['主卡','副卡']

        values=[0, 0]

        axis.bar(names,values)

        output = io.BytesIO()

        FigureCanvasAgg(fig).print\_png(output)

        return Response(output.getvalue(), mimetype="image/png")

    if len(result)==1 and len(prime)==1:

        names=['主卡']

        values=[float(result[0][0])]

        axis.bar(names,values)

        output = io.BytesIO()

        FigureCanvasAgg(fig).print\_png(output)

        return Response(output.getvalue(), mimetype="image/png")

    if len(result)==1 and len(vice)==1:

        names=['副卡']

        values=[float(result[1][0])]

        axis.bar(names,values)

        output = io.BytesIO()

        FigureCanvasAgg(fig).print\_png(output)

        return Response(output.getvalue(), mimetype="image/png")

    names=['主卡','副卡']

    values=[float(result[0][0]),float(result[1][0])]

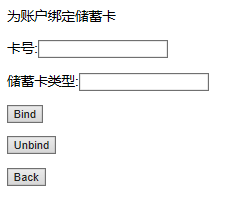
    axis.bar(names,values)

    output = io.BytesIO()

    FigureCanvasAgg(fig).print\_png(output)

    return Response(output.getvalue(), mimetype="image/png")

3.2.11a Bind.html



# 绑定/解绑银行卡

@app.route('/<ID>/<account>/Bind',methods=['POST','GET'])

def Bind(ID,account):

    if ID not in permission.keys():

        permission[ID]=0

    if permission[ID]==0:

        return "请先登录再进行此操作"

    if request.method == 'GET':

        return render\_template('Bind.html',ID=ID,account=account)

    if request.method =='POST':

        if 'bind' in request.form.keys():

            message=""

            card\_num=request.form['cardnum']

            card\_type=request.form['cardtype']

            if card\_num == "" or card\_type == "":

                message="错误：储蓄卡信息不完整"

                return render\_template('Bind.html',ID=ID,account=account,message=message)

            A='''

            select \* from Account\_Card

            where card\_num='{card\_num}' and card\_type='{card\_type}'

            '''.format(card\_num=card\_num,card\_type=card\_type)

            result=DB.ExecQuery(A)

            if len(result)!=0:

                message='错误：储蓄卡已被绑定'

                return render\_template('Bind.html',ID=ID,account=account,message=message)

            B='''

            select count(\*) from Account\_Card

            where account='{account}' and card\_num='{card\_num}' and card\_type='{card\_type}'

            '''.format(account=account,card\_num=card\_num,card\_type=card\_type)

            result=DB.ExecQuery(B)

            if int(result[0][0])!=0:

                message='Error: Card of this type Has Been Bound'

                return render\_template('Bind.html',ID=ID,account=account,message=message)

            if message=="":

                A='''

                insert

                into

                Account\_Card

                values('{account}','{card\_num}','{card\_type}','0')

                '''.format(account=account,card\_num=card\_num,card\_type=card\_type)

                DB.ExecNonQuery(A)

                B='''

                insert

                into

                Trade\_History(ID,summary,my\_account,sb\_account,card\_type,operation,money,pre\_balance,post\_balance,card\_num)

                values('{ID}','用户建立账户绑定银行卡','{account}',' ','{card\_type}','绑定储蓄卡','0','0','0','{card\_num}')

                '''.format(ID=ID,account=account,card\_type=card\_type,card\_num=card\_num)

                DB.ExecNonQuery(B)

                return redirect(url\_for('Bind',ID=ID,account=account,message=message))

        if 'unbind' in request.form.keys():

            message=""

            card\_num=request.form['cardnum']

            card\_type=request.form['cardtype']

            if card\_num == "" or card\_type == "":

                message="错误：储蓄卡信息不完整"

                return render\_template('Bind.html',ID=ID,account=account,message=message)

            A='''

            select \* from Account\_Card

            where account='{account}' and card\_num='{card\_num}' and card\_type='{card\_type}'

            '''.format(account=account,card\_num=card\_num,card\_type=card\_type)

            result=DB.ExecQuery(A)

            if len(result)==0:

                message='错误：储蓄卡已被绑定'

                return render\_template('Bind.html',ID=ID,account=account,message=message)

            if message=="":

                A='''

                delete

                from

                Account\_Card

                where account ='{account}'

                and card\_num ='{card\_num}'

                and card\_type ='{card\_type}'

                '''.format(account=account,card\_num=card\_num,card\_type=card\_type)

                DB.ExecNonQuery(A)

                B='''

                insert

                into

                Trade\_History(ID,summary,my\_account,sb\_account,card\_type,operation,money,pre\_balance,post\_balance,card\_num)

                values('{ID}','用户账户取消绑定银行卡','{account}',' ','{card\_type}','解绑储蓄卡','0','0','0','{card\_num}')

                '''.format(ID=ID,account=account,card\_type=card\_type,card\_num=card\_num)

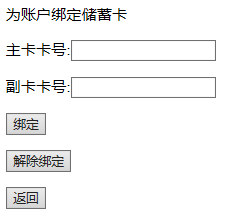
                DB.ExecNonQuery(B)

                return redirect(url\_for('account\_center',ID=ID,account=account,message=message))

        if 'back' in request.form.keys():

            return redirect(url\_for('account\_center',ID=ID,account=account))

3.2.11b Bind2.html



# 绑定/解绑两张银行卡

@app.route('/<ID>/<account>/Bind2',methods=['POST','GET'])

def Bind2(ID,account):

    if ID not in permission.keys():

        permission[ID]=0

    if permission[ID]==0:

        return "请先登录再进行此操作"

    if request.method == 'GET':

        return render\_template('Bind2.html',ID=ID,account=account)

    if request.method =='POST':

        if 'bind' in request.form.keys():

            message=""

            card\_num1=request.form['cardnum1']

            card\_num2=request.form['cardnum2']

            if card\_num1 == "" and card\_num2 == "":

                message="错误：请输入储蓄卡卡号"

                return render\_template('Bind2.html',ID=ID,account=account,message=message)

            if card\_num2 == "":

                # 仅填写了主卡信息

                A='''

                select \* from Account\_Card

                where card\_num='{card\_num1}' and card\_type='主卡'

                '''.format(card\_num1=card\_num1)

                result=DB.ExecQuery(A)

                if len(result)!=0:

                    message='错误：主卡已被绑定'

                    return render\_template('Bind2.html',ID=ID,account=account,message=message)

                B='''

                select count(\*) from Account\_Card

                where account='{account}' and card\_num='{card\_num1}' and card\_type='主卡'

                '''.format(account=account,card\_num1=card\_num1)

                result=DB.ExecQuery(B)

                if int(result[0][0])!=0:

                    message='错误：账户已绑定主卡'

                    return render\_template('Bind2.html',ID=ID,account=account,message=message)

                if message=="":

                    C='''

                    insert

                    into

                    Account\_Card

                    values('{account}','{card\_num1}','主卡','0')

                    '''.format(account=account,card\_num1=card\_num1)

                    DB.ExecNonQuery(C)

                    D='''

                    insert

                    into

                    Trade\_History(ID,summary,my\_account,sb\_account,card\_type,operation,money,pre\_balance,post\_balance,card\_num)

                    values('{ID}','用户建立账户绑定银行卡','{account}',' ','主卡','绑定储蓄卡','0','0','0','{card\_num1}')

                    '''.format(ID=ID,account=account,card\_num1=card\_num1)

                    DB.ExecNonQuery(D)

                    return redirect(url\_for('Bind2',ID=ID,account=account,message=message))

            elif card\_num1 == "":

                # 仅填写了副卡信息

                A='''

                select \* from Account\_Card

                where card\_num='{card\_num2}' and card\_type='副卡'

                '''.format(card\_num2=card\_num2)

                result=DB.ExecQuery(A)

                if len(result)!=0:

                    message='错误：副卡已被绑定'

                    return render\_template('Bind2.html',ID=ID,account=account,message=message)

                B='''

                select count(\*) from Account\_Card

                where account='{account}' and card\_num='{card\_num2}' and card\_type='副卡'

                '''.format(account=account,card\_num2=card\_num2)

                result=DB.ExecQuery(B)

                if int(result[0][0])!=0:

                    message='错误：账户已绑定副卡'

                    return render\_template('Bind2.html',ID=ID,account=account,message=message)

                if message=="":

                    C='''

                    insert

                    into

                    Account\_Card

                    values('{account}','{card\_num2}','副卡','0')

                    '''.format(account=account,card\_num2=card\_num2)

                    DB.ExecNonQuery(C)

                    D='''

                    insert

                    into

                    Trade\_History(ID,summary,my\_account,sb\_account,card\_type,operation,money,pre\_balance,post\_balance,card\_num)

                    values('{ID}','用户建立账户绑定银行卡','{account}',' ','副卡','绑定储蓄卡','0','0','0','{card\_num2}')

                    '''.format(ID=ID,account=account,card\_num2=card\_num2)

                    DB.ExecNonQuery(D)

                    return redirect(url\_for('Bind2',ID=ID,account=account,message=message))

            else:

                # 填写了主卡副卡信息

                # 检查主卡

                A1='''

                select \* from Account\_Card

                where card\_num='{card\_num1}' and card\_type='主卡'

                '''.format(card\_num1=card\_num1)

                result=DB.ExecQuery(A1)

                if len(result)!=0:

                    message='错误：主卡已被绑定'

                    return render\_template('Bind2.html',ID=ID,account=account,message=message)

                B1='''

                select count(\*) from Account\_Card

                where account='{account}' and card\_num='{card\_num1}' and card\_type='主卡'

                '''.format(account=account,card\_num1=card\_num1)

                result=DB.ExecQuery(B1)

                if int(result[0][0])!=0:

                    message='错误：账户已绑定主卡'

                    return render\_template('Bind2.html',ID=ID,account=account,message=message)

                if message=="":

                    C1='''

                    insert

                    into

                    Account\_Card

                    values('{account}','{card\_num1}','主卡','0')

                    '''.format(account=account,card\_num1=card\_num1)

                    DB.ExecNonQuery(C1)

                    D1='''

                    insert

                    into

                    Trade\_History(ID,summary,my\_account,sb\_account,card\_type,operation,money,pre\_balance,post\_balance,card\_num)

                    values('{ID}','用户建立账户绑定银行卡','{account}',' ','主卡','绑定储蓄卡','0','0','0','{card\_num1}')

                    '''.format(ID=ID,account=account,card\_num1=card\_num1)

                    DB.ExecNonQuery(D1)

                # 检查副卡

                A2='''

                select \* from Account\_Card

                where card\_num='{card\_num2}' and card\_type='副卡'

                '''.format(card\_num2=card\_num2)

                result=DB.ExecQuery(A2)

                if len(result)!=0:

                    message='错误：副卡已被绑定'

                    return render\_template('Bind2.html',ID=ID,account=account,message=message)

                B2='''

                select count(\*) from Account\_Card

                where account='{account}' and card\_num='{card\_num2}' and card\_type='副卡'

                '''.format(account=account,card\_num2=card\_num2)

                result=DB.ExecQuery(B2)

                if int(result[0][0])!=0:

                    message='错误：账户已绑定副卡'

                    return render\_template('Bind2.html',ID=ID,account=account,message=message)

                if message=="":

                    C2='''

                    insert

                    into

                    Account\_Card

                    values('{account}','{card\_num2}','副卡','0')

                    '''.format(account=account,card\_num2=card\_num2)

                    DB.ExecNonQuery(C2)

                    D2='''

                    insert

                    into

                    Trade\_History(ID,summary,my\_account,sb\_account,card\_type,operation,money,pre\_balance,post\_balance,card\_num)

                    values('{ID}','用户建立账户绑定银行卡','{account}',' ','副卡','绑定储蓄卡','0','0','0','{card\_num2}')

                    '''.format(ID=ID,account=account,card\_num2=card\_num2)

                    DB.ExecNonQuery(D2)

                    return redirect(url\_for('Bind2',ID=ID,account=account,message=message))

        if 'unbind' in request.form.keys():

            message=""

            card\_num1=request.form['cardnum1']

            card\_num2=request.form['cardnum2']

            if card\_num1 == "" and card\_num2 == "":

                message="错误：卡号信息不完整"

                return render\_template('Bind2.html',ID=ID,account=account,message=message)

            if card\_num2 == "":

                # 解绑主卡

                A='''

                select \* from Account\_Card

                where account='{account}' and card\_num='{card\_num1}' and card\_type='主卡'

                '''.format(account=account,card\_num1=card\_num1)

                result=DB.ExecQuery(A)

                if len(result)==0:

                    message='错误：主卡未被绑定'

                    return render\_template('Bind2.html',ID=ID,account=account,message=message)

                if message=="":

                    A='''

                    delete

                    from

                    Account\_Card

                    where account ='{account}'

                    and card\_num ='{card\_num1}'

                    and card\_type ='主卡'

                    '''.format(account=account,card\_num1=card\_num1)

                    DB.ExecNonQuery(A)

                    B='''

                    insert

                    into

                    Trade\_History(ID,summary,my\_account,sb\_account,card\_type,operation,money,pre\_balance,post\_balance,card\_num)

                    values('{ID}','用户账户取消绑定银行卡','{account}',' ','主卡','解绑储蓄卡','0','0','0','{card\_num1}')

                    '''.format(ID=ID,account=account,card\_num1=card\_num1)

                    DB.ExecNonQuery(B)

                return redirect(url\_for('account\_center',ID=ID,account=account,message=message))

            elif card\_num1 == "":

                # 解绑副卡

                A='''

                select \* from Account\_Card

                where account='{account}' and card\_num='{card\_num2}' and card\_type='副卡'

                '''.format(account=account,card\_num2=card\_num2)

                result=DB.ExecQuery(A)

                if len(result)==0:

                    message='错误：副卡未被绑定'

                    return render\_template('Bind2.html',ID=ID,account=account,message=message)

                if message=="":

                    A='''

                    delete

                    from

                    Account\_Card

                    where account ='{account}'

                    and card\_num ='{card\_num2}'

                    and card\_type ='副卡'

                    '''.format(account=account,card\_num2=card\_num2)

                    DB.ExecNonQuery(A)

                    B='''

                    insert

                    into

                    Trade\_History(ID,summary,my\_account,sb\_account,card\_type,operation,money,pre\_balance,post\_balance,card\_num)

                    values('{ID}','用户账户取消绑定银行卡','{account}',' ','副卡','解绑储蓄卡','0','0','0','{card\_num2}')

                    '''.format(ID=ID,account=account,card\_num2=card\_num2)

                    DB.ExecNonQuery(B)

                return redirect(url\_for('account\_center',ID=ID,account=account,message=message))

            else:

                # 解绑主卡与副卡

                A1='''

                select \* from Account\_Card

                where account='{account}' and card\_num='{card\_num1}' and card\_type='主卡'

                '''.format(account=account,card\_num1=card\_num1)

                result=DB.ExecQuery(A1)

                if len(result)==0:

                    message='错误：主卡未被绑定'

                    return render\_template('Bind2.html',ID=ID,account=account,message=message)

                if message=="":

                    A1='''

                    delete

                    from

                    Account\_Card

                    where account ='{account}'

                    and card\_num ='{card\_num1}'

                    and card\_type ='主卡'

                    '''.format(account=account,card\_num1=card\_num1)

                    DB.ExecNonQuery(A1)

                    B1='''

                    insert

                    into

                    Trade\_History(ID,summary,my\_account,sb\_account,card\_type,operation,money,pre\_balance,post\_balance,card\_num)

                    values('{ID}','用户账户取消绑定银行卡','{account}',' ','主卡','解绑储蓄卡','0','0','0','{card\_num1}')

                    '''.format(ID=ID,account=account,card\_num1=card\_num1)

                    DB.ExecNonQuery(B1)

                A2='''

                select \* from Account\_Card

                where account='{account}' and card\_num='{card\_num2}' and card\_type='副卡'

                '''.format(account=account,card\_num2=card\_num2)

                result=DB.ExecQuery(A2)

                if len(result)==0:

                    message='错误：副卡未被绑定'

                    return render\_template('Bind2.html',ID=ID,account=account,message=message)

                if message=="":

                    A2='''

                    delete

                    from

                    Account\_Card

                    where account ='{account}'

                    and card\_num ='{card\_num2}'

                    and card\_type ='副卡'

                    '''.format(account=account,card\_num2=card\_num2)

                    DB.ExecNonQuery(A2)

                    B3='''

                    insert

                    into

                    Trade\_History(ID,summary,my\_account,sb\_account,card\_type,operation,money,pre\_balance,post\_balance,card\_num)

                    values('{ID}','用户账户取消绑定银行卡','{account}',' ','副卡','解绑储蓄卡','0','0','0','{card\_num2}')

                    '''.format(ID=ID,account=account,card\_num2=card\_num2)

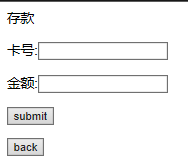
                    DB.ExecNonQuery(B3)

            return redirect(url\_for('account\_center',ID=ID,account=account))

        if 'back' in request.form.keys():

            return redirect(url\_for('account\_center',ID=ID,account=account))

3.2.12 Deposit.html



# 存钱

@app.route('/<ID>/<account>/Deposit',methods=['POST','GET'])

def Deposit(ID,account):

    if ID not in permission.keys():

        permission[ID]=0

    if permission[ID]==0:

        return "请先登录再进行此操作"

    if request.method == 'GET':

        return render\_template('Deposit.html',ID=ID,account=account)

    if request.method =='POST':

        if 'submit' in request.form.keys():

            message=""

            card\_num=request.form['cardnum']

            money=request.form['money']

            if card\_num == "" or money == "":

                message="错误：存款信息不完整"

                return render\_template('Deposit.html',ID=ID,account=account,message=message)

            else:

                A='''

                select \* from Account\_Card

                where account='{account}' and card\_num='{card\_num}'

                '''.format(account=account,card\_num=card\_num)

                result=DB.ExecQuery(A)

                if len(result)==0:

                    message="错误：储蓄卡未绑定账户"

                    return render\_template('Deposit.html',ID=ID,account=account,message=message)

                else:

                    card\_type=result[0][2]

                    pre\_balance=result[0][3]

                    B='''

                    update Account\_Card

                    set balance = balance + '{money}'

                    where account = '{account}' and card\_num = '{card\_num}'

                    '''.format(money=money,account=account,card\_num=card\_num)

                    DB.ExecNonQuery(B)

                    C='''

                    select balance from Account\_Card

                    where account='{account}' and card\_num='{card\_num}'

                    '''.format(account=account,card\_num=card\_num)

                    result=DB.ExecQuery(C)

                    post\_balance=result[0][0]

                    D='''

                    insert

                    into

                    Trade\_History(ID,summary,my\_account,sb\_account,card\_type,operation,money,pre\_balance,post\_balance,card\_num)

                    values('{ID}','用户账户存款','{account}',' ','{card\_type}','存款','{money}','{pre\_balance}','{post\_balance}','{card\_num}')

                    '''.format(ID=ID,account=account,card\_type=card\_type,money=money,pre\_balance=pre\_balance,post\_balance=post\_balance,card\_num=card\_num)

                    DB.ExecNonQuery(D)

                    E='''

                    update Account

                    set Account.balance = (select sum(balance)

                                          from Account\_Card

                                          where Account\_Card.account = '{account}'

                                          )

                    where account = '{account}'

                    '''.format(account=account)

                    DB.ExecNonQuery(E)

                    # 从第一笔存款开始计息

                    F='''

                    select \* from Cal\_Interest

                    where account = '{account}'

                    '''.format(account=account)

                    result=DB.ExecQuery(F)

                    if len(result)==0:

                        G='''

                        insert

                        into

                        Cal\_Interest

                        values('{account}','','','0')

                        '''.format(account=account)

                        DB.ExecNonQuery(G)

                        H='''

                        update Cal\_Interest

                        set cur\_calc = GETDATE(),pre\_calc = GETDATE()

                        where account = '{account}'

                        '''.format(account=account)

                        DB.ExecNonQuery(H)

                    else:

                        # 计息函数

                        calc\_interest(account,pre\_balance)

                    return redirect(url\_for('account\_center',ID=ID,account=account,message=message))

        if 'back' in request.form.keys():

            return redirect(url\_for('account\_center',ID=ID,account=account))

3.2.13 Withdraw.html



# 取钱

@app.route('/<ID>/<account>/Withdraw',methods=['POST','GET'])

def Withdraw(ID,account):

    if ID not in permission.keys():

        permission[ID]=0

    if permission[ID]==0:

        return "请先登录再进行此操作"

    if request.method == 'GET':

        return render\_template('Withdraw.html',ID=ID,account=account)

    if request.method =='POST':

        if 'submit' in request.form.keys():

            message=""

            card\_num=request.form['cardnum']

            money=request.form['money']

            if card\_num == "" or money == "":

                message="错误：取款信息不完整"

                return render\_template('Withdraw.html',ID=ID,account=account,message=message)

            else:

                A='''

                select \* from Account\_Card

                where account='{account}' and card\_num='{card\_num}'

                '''.format(account=account,card\_num=card\_num)

                result=DB.ExecQuery(A)

                if len(result)==0:

                    message="错误：储蓄卡未绑定账户"

                    return render\_template('Withdraw.html',ID=ID,account=account,message=message)

                else:

                    B='''

                    select \* from Account\_Card

                    where account='{account}' and card\_num='{card\_num}' and balance>'{money}'

                    '''.format(account=account,card\_num=card\_num,money=money)

                    result=DB.ExecQuery(B)

                    if len(result)==0:

                        message="错误：余额不足"

                        return render\_template('Withdraw.html',ID=ID,account=account,message=message)

                    else:

                        C='''

                        select \* from Account\_Card

                        where account='{account}' and card\_num='{card\_num}'

                        '''.format(account=account,card\_num=card\_num)

                        result=DB.ExecQuery(C)

                        card\_type=result[0][2]

                        pre\_balance=result[0][3]

                        D='''

                        update Account\_Card

                        set balance = balance - '{money}'

                        where account = '{account}' and card\_num = '{card\_num}'

                        '''.format(money=money,account=account,card\_num=card\_num)

                        DB.ExecNonQuery(D)

                        E='''

                        select balance from Account\_Card

                        where account='{account}' and card\_num='{card\_num}'

                        '''.format(account=account,card\_num=card\_num)

                        result=DB.ExecQuery(E)

                        post\_balance=result[0][0]

                        F='''

                        insert

                        into

                        Trade\_History(ID,summary,my\_account,sb\_account,card\_type,operation,money,pre\_balance,post\_balance,card\_num)

                        values('{ID}','用户账户取款','{account}',' ','{card\_type}','取款','{money}','{pre\_balance}','{post\_balance}','{card\_num}')

                        '''.format(ID=ID,account=account,card\_type=card\_type,money=money,pre\_balance=pre\_balance,post\_balance=post\_balance,card\_num=card\_num)

                        DB.ExecNonQuery(F)

                        G='''

                        update Account

                        set Account.balance = (select sum(balance)

                                            from Account\_Card

                                            where Account\_Card.account = '{account}'

                                            )

                        where account = '{account}'

                        '''.format(account=account)

                        DB.ExecNonQuery(G)

                        # 计息函数

                        calc\_interest(account,pre\_balance)

                        return redirect(url\_for('account\_center',ID=ID,account=account,message=message))

        if 'back' in request.form.keys():

            return redirect(url\_for('account\_center',ID=ID,account=account))

3.2.14 Transfer.html



# 转账

@app.route('/<ID>/<account>/Transfer',methods=['POST','GET'])

def Transfer(ID,account):

    if ID not in permission.keys():

        permission[ID]=0

    if permission[ID]==0:

        return "请先登录再进行此操作"

    if request.method == 'GET':

        return render\_template('Transfer.html',ID=ID,account=account)

    if request.method =='POST':

        if 'submit' in request.form.keys():

            message=""

            mycard\_num=request.form['mycardnum']

            sbcard\_num=request.form['sbcardnum']

            money=request.form['money']

            if mycard\_num == "" or sbcard\_num =="" or money == "":

                message="错误：转账信息不完整"

                return render\_template('Transfer.html',ID=ID,account=account,message=message)

            else:

                A='''

                select \* from Account\_Card

                where account='{account}' and card\_num='{card\_num}'

                '''.format(account=account,card\_num=mycard\_num)

                result=DB.ExecQuery(A)

                if len(result)==0:

                    message="错误：您的储蓄卡未绑定账户"

                    return render\_template('Transfer.html',ID=ID,account=account,message=message)

                B='''

                select Account\_Card.account,Account.ID

                from Account\_Card

                inner join Account

                on Account\_Card.account = Account.account

                where card\_num='{card\_num}'

                '''.format(card\_num=sbcard\_num)

                result=DB.ExecQuery(B)

                if len(result)==0:

                    message="错误：对方储蓄卡未绑定账户"

                    return render\_template('Transfer.html',ID=ID,account=account,message=message)

                sb\_account=result[0][0]

                sb\_ID=result[0][1]

                C='''

                select \* from Account\_Card

                where account='{account}' and card\_num='{card\_num}' and balance>'{money}'

                '''.format(account=account,card\_num=mycard\_num,money=money)

                result=DB.ExecQuery(C)

                if len(result)==0:

                    message="错误：您的储蓄卡余额不足"

                    return render\_template('Transfer.html',ID=ID,account=account,message=message)

                # 从本人卡上扣钱

                D='''

                select \* from Account\_Card

                where account='{account}' and card\_num='{card\_num}'

                '''.format(account=account,card\_num=mycard\_num)

                result=DB.ExecQuery(D)

                mycard\_type=result[0][2]

                mypre\_balance=result[0][3]

                E='''

                update Account\_Card

                set balance = balance - '{money}'

                where account = '{account}' and card\_num = '{card\_num}'

                '''.format(money=money,account=account,card\_num=mycard\_num)

                DB.ExecNonQuery(E)

                F='''

                select balance from Account\_Card

                where account='{account}' and card\_num='{card\_num}'

                '''.format(account=account,card\_num=mycard\_num)

                result= DB.ExecQuery(F)

                mypost\_balance=result[0][0]

                G='''

                insert

                into

                Trade\_History(ID,summary,my\_account,sb\_account,card\_type,operation,money,pre\_balance,post\_balance,card\_num)

                values('{ID}','用户向其他账户转账','{myaccount}','{sbaccount}','{card\_type}','发起转账','{money}','{pre\_balance}','{post\_balance}','{card\_num}')

                '''.format(ID=ID,myaccount=account,sbaccount=sb\_account,card\_type=mycard\_type,money=money,pre\_balance=mypre\_balance,post\_balance=mypost\_balance,card\_num=mycard\_num)

                DB.ExecNonQuery(G)

                G1='''

                update Account

                set Account.balance = (select sum(balance)

                                      from Account\_Card

                                      where Account\_Card.account = '{account}'

                                      )

                where account = '{account}'

                '''.format(account=account)

                DB.ExecNonQuery(G1)

                # 调用计息函数

                calc\_interest(account,mypre\_balance)

                # 收到转账

                H='''

                select \* from Account\_Card

                where account='{account}' and card\_num='{card\_num}'

                '''.format(account=sb\_account,card\_num=sbcard\_num)

                result=DB.ExecQuery(H)

                sbcard\_type=result[0][2]

                sbpre\_balance=result[0][3]

                I='''

                update Account\_Card

                set balance = balance + '{money}'

                where account = '{account}' and card\_num = '{card\_num}'

                '''.format(money=money,account=sb\_account,card\_num=sbcard\_num)

                DB.ExecNonQuery(I)

                J='''

                select balance from Account\_Card

                where account='{account}' and card\_num='{card\_num}'

                '''.format(account=sb\_account,card\_num=sbcard\_num)

                result=DB.ExecQuery(J)

                sbpost\_balance=result[0][0]

                K='''

                insert

                into

                Trade\_History(ID,summary,my\_account,sb\_account,card\_type,operation,money,pre\_balance,post\_balance,card\_num)

                values('{ID}','用户收到其他账户转账','{account}','{sbaccount}','{card\_type}','收到转账','{money}','{pre\_balance}','{post\_balance}','{card\_num}')

                '''.format(ID=sb\_ID,account=sb\_account,sbaccount=account,card\_type=sbcard\_type,money=money,pre\_balance=sbpre\_balance,post\_balance=sbpost\_balance,card\_num=sbcard\_num)

                DB.ExecNonQuery(K)

                K1='''

                update Account

                set Account.balance = (select sum(balance)

                                      from Account\_Card

                                      where Account\_Card.account = '{account}'

                                      )

                where account = '{account}'

                '''.format(account=sb\_account)

                DB.ExecNonQuery(K1)

                # 调用计息函数

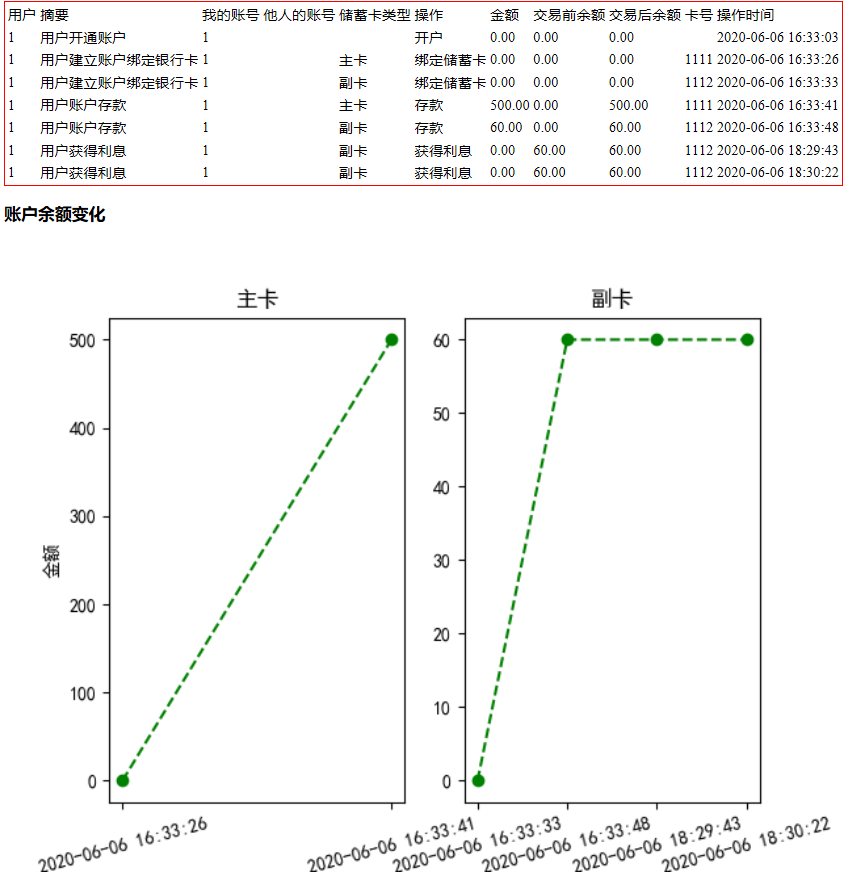
                calc\_interest(sb\_account,sbpre\_balance)

                return redirect(url\_for('account\_center',ID=ID,account=account,message=message))

        if 'back' in request.form.keys():

            return redirect(url\_for('account\_center',ID=ID,account=account))

3.2.15 查询交易历史记录



# 显示账户余额变化

@app.route("/balance\_variation/<int:ID>/<int:account>.png")

def variation\_plot\_png(ID,account):

    fig = Figure()

    axis1 = fig.add\_subplot(1, 2, 1)

    axis2 = fig.add\_subplot(1, 2, 2)

    axis1.set\_title('主卡')

    axis2.set\_title('副卡')

    axis1.set\_ylabel('金额')

    names1=[]

    values1=[]

    A='''

    select post\_balance,OrderDate from Trade\_History

    where ID ='{ID}' and my\_account='{account}' and card\_type = '主卡'

    '''.format(ID=ID,account=account)

    result1=DB.ExecQuery(A)

    for \_ in range(len(result1)):

        names1.append((str(result1[\_][1]))[:19])

        values1.append(float(result1[\_][0]))

    for tick in axis1.get\_xticklabels():  # 将横坐标倾斜30度，纵坐标可用相同方法

        tick.set\_rotation(15)

    axis1.plot(names1,values1,'go--')

    names2=[]

    values2=[]

    B='''

    select post\_balance,OrderDate from Trade\_History

    where ID ='{ID}' and my\_account='{account}' and card\_type = '副卡'

    '''.format(ID=ID,account=account)

    result2=DB.ExecQuery(B)

    for \_ in range(len(result2)):

        names2.append((str(result2[\_][1]))[:19])

        values2.append(float(result2[\_][0]))

    for tick in axis2.get\_xticklabels():  # 将横坐标倾斜30度，纵坐标可用相同方法

        tick.set\_rotation(15)

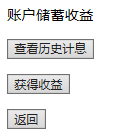
    axis2.plot(names2,values2,'go--')

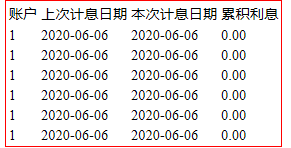
    output = io.BytesIO()

    FigureCanvasAgg(fig).print\_png(output)

    return Response(output.getvalue(), mimetype="image/png")

3.2.16 查询计息历史记录





# 账户利息中心

@app.route('/<ID>/<account>/Interest',methods=['POST','GET'])

def Interest(ID,account):

    if ID not in permission.keys():

        permission[ID]=0

    if permission[ID]==0:

        return "请先登录再进行此操作"

    if request.method == 'GET':

        return render\_template('Interest.html',ID=ID,account=account)

    if request.method =='POST':

        if 'history' in request.form.keys():

            message=""

            A='''

            select \* from Cal\_Interest

            where account = '{account}'

            '''.format(account=account)

            result=DB.ExecQuery(A)

            if len(result)==0:

                message='账户尚未存入现金'

                return render\_template('Interest.html',ID=ID,account=account)

            if message=="":

                cursor = DB.GetConnect()

                s = "<table style='border:1px solid red'>"

                s = s + "<tr><td>账户</td><td>上次计息日期</td><td>本次计息日期</td><td>累积利息</td></tr>"

                with cursor.execute(A):

                    row = cursor.fetchone()

                    while row:

                        s = s + "<tr>"

                        s = s + "<td>" + str(row[0]) + "</td>"

                        print(str(row[0]))

                        s = s + "<td>" + str(row[1]) + "</td>"

                        print(str(row[1]))

                        s = s + "<td>" + str(row[2]) + "</td>"

                        print(str(row[2]))

                        s = s + "<td>" + str(row[3]) + "</td>"

                        print(str(row[3]))

                        row = cursor.fetchone()

                        s = s +  "</tr>"

                s = s + "</table>"

                s = "<html><body>" + s +"</body></html>"

                return render\_template\_string(s,ID=ID,account=account)

        if 'extract' in request.form.keys():

            message=""

            A='''

            select \* from Cal\_Interest

            where account = '{account}'

            '''.format(account=account)

            result=DB.ExecQuery(A)

            if len(result)==0:

                message='账户尚未存入现金'

                return render\_template('account\_center.html',ID=ID,account=account)

            if message=="":

                # 添加计息记录

                A='''

                select top 1 interest

                from Cal\_Interest

                where account = '{account}'

                order by cur\_calc desc

                '''.format(account=account)

                result=DB.ExecQuery(A)

                interest=float(result[0][0])

                B='''

                insert

                into

                Cal\_Interest

                values('{account}','','',NULL)

                '''.format(account=account)

                DB.ExecNonQuery(B)

                C='''

                update Cal\_Interest

                set cur\_calc = GETDATE(),pre\_calc = (select max(cur\_calc)

                                                    from Cal\_Interest

                                                    where account = '{account}' and interest is not null

                                                    )

                where account = '{account}'

                '''.format(account=account)

                DB.ExecNonQuery(C)

                D='''

                update Cal\_Interest

                set interest = '0'

                where account = '{account}' and interest is null

                '''.format(interest=interest,account=account)

                DB.ExecNonQuery(D)

                # 利息转入某卡中

                E='''

                select top 1 post\_balance,card\_num,card\_type

                from Trade\_History

                where ID = '{ID}' and my\_account = '{account}' and operation != '解绑储蓄卡'

                order by OrderDate desc

                '''.format(ID=ID,account=account)

                result=DB.ExecQuery(E)

                pre\_balance = float(result[0][0])

                post\_balance = pre\_balance + interest

                card\_num = int(result[0][1])

                card\_type = str(result[0][2])

                F='''

                insert

                into

                Trade\_History(ID,summary,my\_account,sb\_account,card\_type,operation,money,pre\_balance,post\_balance,card\_num)

                values('{ID}','用户获得利息','{account}',' ','{card\_type}','获得利息','{money}','{pre\_balance}','{post\_balance}','{card\_num}')

                '''.format(ID=ID,account=account,card\_type=card\_type,money=interest,pre\_balance=pre\_balance,post\_balance=post\_balance,card\_num=card\_num)

                DB.ExecNonQuery(F)

                G='''

                update Account

                set Account.balance = (select sum(balance)

                                      from Account\_Card

                                      where Account\_Card.account = '{account}'

                                      )

                where account = '{account}'

                '''.format(account=account)

                DB.ExecNonQuery(G)

                return redirect(url\_for('account\_center',ID=ID,account=account,message=message))

# 计息函数

def calc\_interest(account,pre\_balance):

    I='''

    insert

    into

    Cal\_Interest

    values('{account}','','',NULL)

    '''.format(account=account)

    DB.ExecNonQuery(I)

    # 获取利率

    OP='''

    select top 1 interest

    from Cal\_Interest

    where account = '{account}'

    order by cur\_calc desc

    '''.format(account=account)

    result=DB.ExecQuery(OP)

    interest=float(result[0][0])

    J='''

    update Cal\_Interest

    set cur\_calc = GETDATE(),pre\_calc = (select max(cur\_calc)

                                        from Cal\_Interest

                                        where account = '{account}' and interest is not null

                                        )

    where account = '{account}'

    '''.format(account=account)

    DB.ExecNonQuery(J)

    K='''

    select datediff(day,pre\_calc,cur\_calc) as daydiff

    from Cal\_Interest

    where interest is null

    '''

    result=DB.ExecQuery(K)

    daydiff=int(result[0][0])

    interest += 0.015\* daydiff /365 \* float(pre\_balance)

    L='''

    update Cal\_Interest

    set interest = '{interest}'

    where account = '{account}' and interest is null

    '''.format(interest=interest,account=account)

    DB.ExecNonQuery(L)

**5 总结**

这次大作业，是我们对现在所学的 《数据库系统概论》 书中理论知识的实践，题目是“银行储蓄业务系统” 。我通过逐步的分析结构，勾画了大致的数据库框架。我经历了需求分析、概念结构设计、逻辑结构设计、物理结构设计、数据库实施、网页app开发等过程，对数据库的安全性、完整性层层把握，完善数据库的设计。

我们在课堂上学习了，数据库编程的相关知识，应用 SQL sever进行数据库设计。通过这次大作业发现这其中需要的很多知识我们没有接触过， 去github查资料的时候发现我们前边所学到的仅仅是皮毛。 同时也发现有很多已经学过的东西我们没有理解到位，不能灵活运用于实际， 不能很好的用来解决问题， 这就需要我们不断的大量的实践，通过不断的自学，不断地发现问题，思考问题，进而解决问题。 在这个过程中我们将深刻理解所学知识， 同时也可以学到不少很实用的东西，此外，相信这样的自学能力我们也将终身受用。

本次作业“银行储蓄业务系统”仍然具有提升空间，尤其是计息系统这一部分。当前的计息系统在每次操作之后都要根据计息整体计算利息，只支持用户自己提取利息，这给用户体验造成负面影响。理想的计息系统应该仅记录计息整体的变化，并且能访问系统时间在每月的固定时间根据计息整体和时间记录计算利息，之后转存到用户账户中去。

**6 参考文献**

[1]刘世峰 .数据库基础与应用 .中央广播电视大学出版社 .2010

[2] 李铃.数据库原理与应用 [M]. 北京中国经济出版社 .2001

[3] 王珊、陈红 .《数据库系统原理教程》 .清华大学出版社 .1998 年 7 月

[4] Flask 教程<https://www.w3cschool.cn/flask/>

[5] Flask 教程<https://www.w3school.com.cn/html/>

[6] Python + MySQL 数据库教程 <https://www.jb51.net/Special/864.html>

[7] github 项目<https://github.com/faded53222/simple-Databse-system-with-html-and-sql-server>