



计算机网络

项目开发报告

题 目  **基于B/S架构的银行业务系统**

学 院  **建筑与环境学院**

专 业  **力学-软件工程交叉学科实验班**

学生姓名  **杨杰**

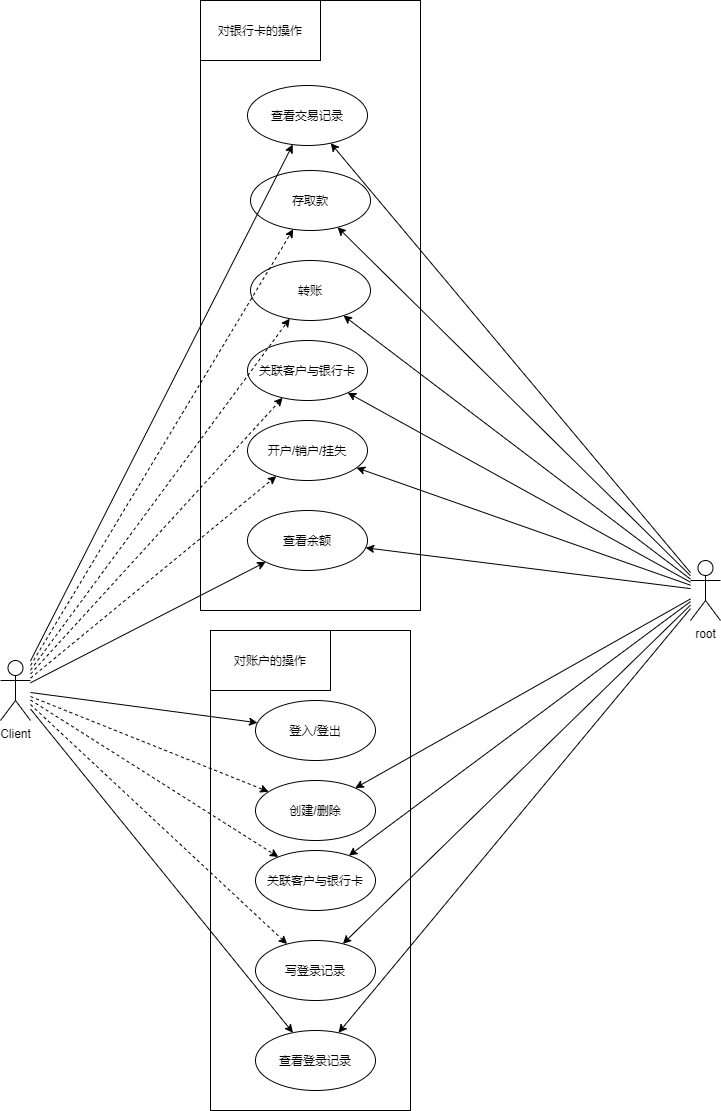
学 号  **2019141470459** 年级 **2019**

指导教师  **宋万忠**

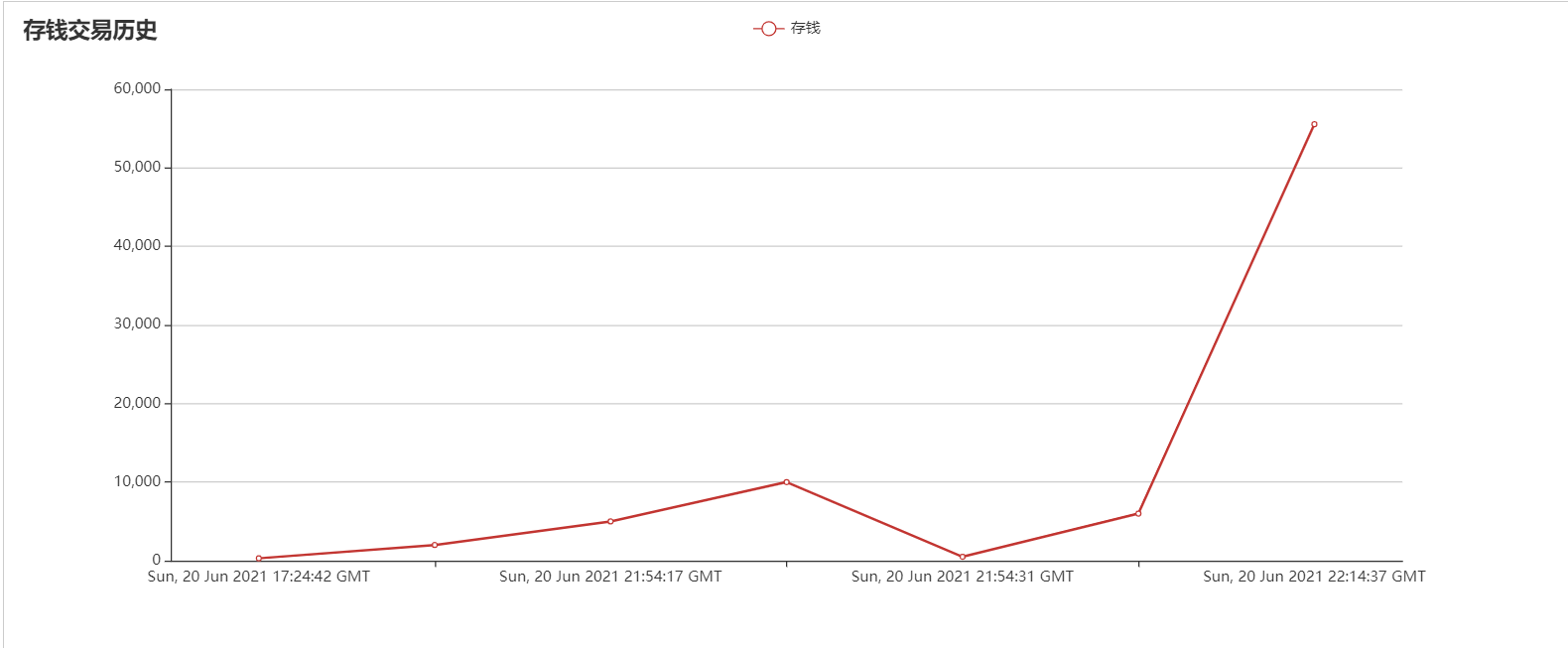
二0二一年 6月 日

目录

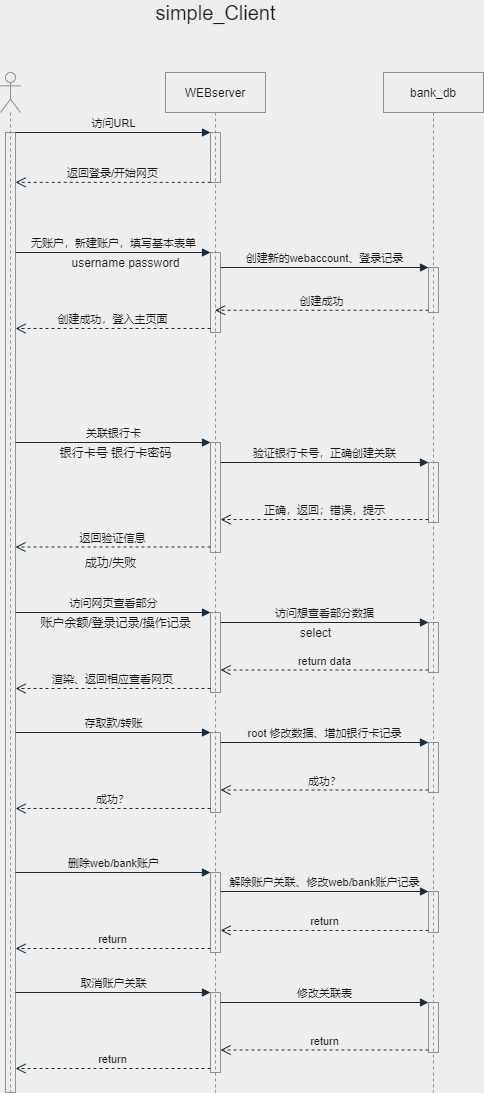
1. **功能及说明**
2. **实现功能的流程及说明**
3. **数据库设计与说明**
4. **项目开发使用的知识点列表**
5. **项目开发中遇到的问题和体会**
6. **附件一：代码**
7. **附件二：视频（PPT）（如果有）**
8. **功能及说明**

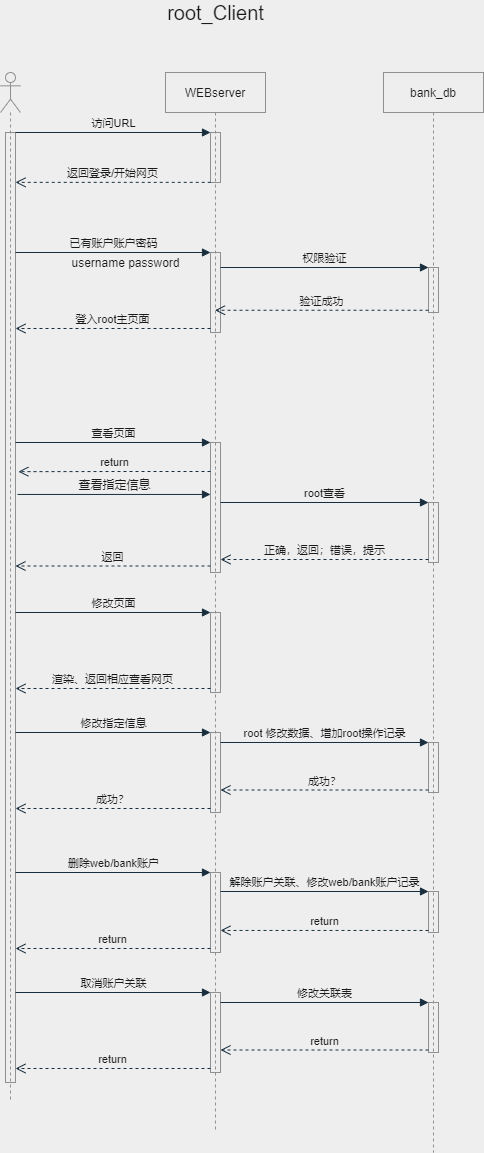
****

1. 建立银行服务器（server）支持多线程web网页访问，用户可以通过浏览器（Browser）通过计算机远程访问服务器办理业务。
2. 建立银行数据库系统建立用户信息，支持不同权限数据库用户增、删、改、查等操作。
3. 用户可以通过远程访问银行web网页完成对账户的查询、转账等功能的操作



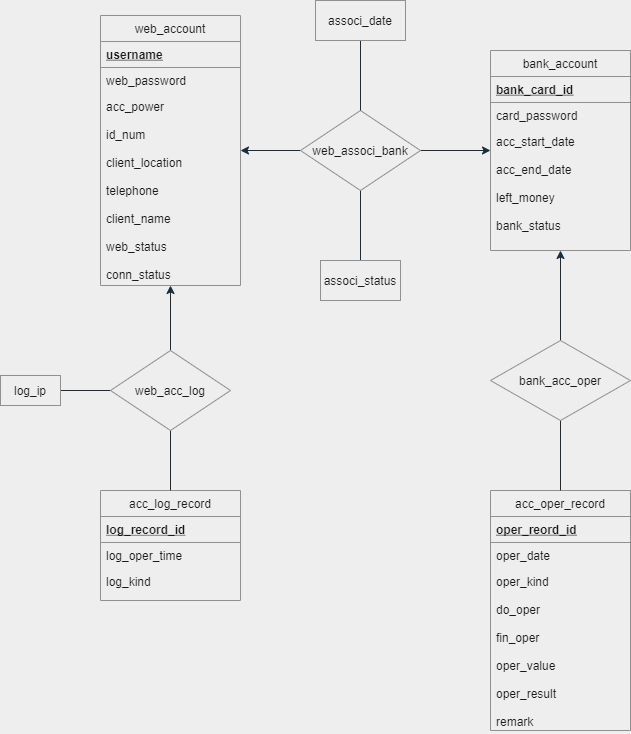
1. 数据库管理员可以有不同操作权限
2. **实现功能的流程及说明**

****

****

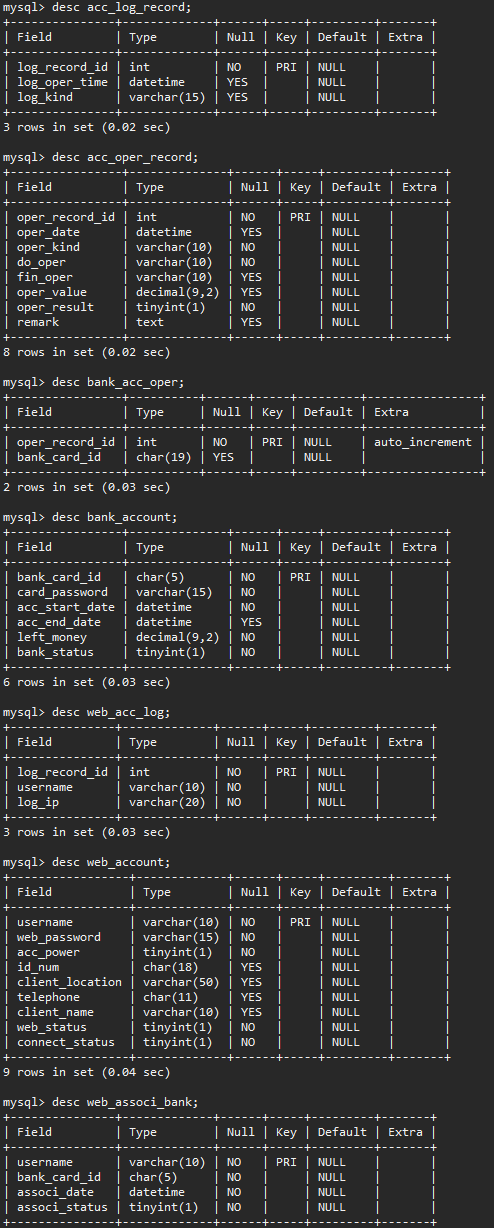
1. 建立银行服务器（server）支持多线程web网页访问，用户可以通过浏览器（Browser）通过计算机远程访问服务器办理业务。
2. 在本地主机通过Python语言和HTML标记语言创建门户web网页和交互程序。
3. 通过python多线程技术允许同时多用户访问网页。
4. 建立银行数据库系统建立用户信息，支持不同权限数据库用户增、删、改、查等操作。
5. 通过数据库设计，事先建立服务器本地数据库。
6. 通过管理员导入、修改用户信息维护服务器数据库。
7. 用户可以通过远程访问银行web网页完成对账户的查询、转账等功能的操作
8. 用户通过账户、密码登录进入个人页操作。
9. 进入个人页，通过选项（关联银行卡、查询账户信息变更流水记录、转账、存入……）。
10. 退出网页。
11. 管理员通过自己的管理员特殊账户于登录页面登入系统，进入管理员页面
12. 管理员通过自己的管理员特殊账户于登录页面登入系统，进入管理员页面
13. 管理员退出登录
14. **数据库设计与说明**

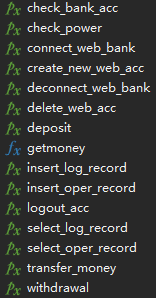
**E-R图：**

****

一共七个关系，即E-R图中所示，实体有四个，分别是网页账户（web\_account）、银行卡账户（bank\_account）、网页登录登出记录（acc\_log\_record）、银行卡账户操作记录（acc\_oper\_record）网页账户与银行卡账户可能存在一一关联的关系，取决于是否绑定银行卡，账户的登录、对账户的操作记录在登入登出的时候自动记录，在关联表（web\_acc\_log、bank\_acc\_oper）中将记录与账户对应。

所有表的字段格式如下





此外实现了一些函数与存储过程：

Check\_bank\_acc：检查银行卡账户是否存在、是否已经被关联

Check\_power：查看网页用户是否存在，如果存在，权限是什么（管理员或是普通用户）

Connect\_web\_bank：关联银行卡账户和网页账户

Create\_new\_web\_acc：创建新网页账户

Deconnect\_web\_bank：取消已有的网页账户和银行卡账户之间的关联关系

Delete\_web\_acc：删除某网页账户

Deposit：给账户存钱

Getmoney：获得当前账户余额

Insert\_log\_record：插入登陆登出操作记录

Insert\_oper\_record：插入操作记录

Logout\_acc：登出账户

Select\_log\_record：筛选登录记录

Select\_oper\_record：筛选操作记录

Transefer\_money：转账

Withdrawal：取钱

1. **项目开发使用的知识点列表**

Socket编程

MySQL与Python自写webserver交互

Python webserver与flask框架的web应用交互

动态获取数据在前端绘制图像（echarts）

数据库设计

E-R图绘制

创建数据库

创建表

创建函数与存储过程

创建用户，赋予权限

1. **项目开发中遇到的问题和解决办法**

问题1：

实现网页的跳转和校核全放在server中显得很臃肿且不易扩展

解决：使用flask框架更容易完成这一工作

问题2：

Server分解的报文体内容在flask框架写的flask\_app中无法使用request请求获取到

解决：全部使用get请求的方式将请求数据放在url后，然后重写get函数（flask\_app.py中我重新写了一个函数叫做get\_values），自己使用正则表达式将其分解处理得到信息，完成了前后端交互

问题3：

选用持久性连接还是非持久性连接

解决：在实际我的项目中，持久性连接并不是一个好选择，因为以下原因：

1. 用户的反应时长完全不可确定，某些页面可能停留极长时间，这会大大增加线程负担
2. Python自带的多线程性能并不好，无法解放计算机性能，其处理能力并不会高出串行很多。

综上，我选择非持久性连接，虽然在对于TCP而言握手挥手的效率降低了，但是整体的资源消耗减少了，或能服务更多连接。

但是如果一定要实现持久性连接也是可以做到的，我可以在我的server中维护一个连接表，新请求会索引这张表如果不存在匹配项则新开线程，如果存在则那个线程不断监听来自同一地址的请求然后返回数据

问题4：

Cookie的实现

解决：我的项目中cookie虽然可以放在报文中，但flask应用并不能找到（同问题2），所以在分解url用多参数，将用户信息摻入其中

1. **附件一：代码**

**Python代码：（server.py）（flask\_app.py请查看源代码）**

import socket

import io

import sys

import datetime

import threading

import configparser

class WSGIServer(object):

    # socket的两个参数初始化

    address\_family = socket.AF\_INET

    socket\_type = socket.SOCK\_STREAM

    # 允许队列，后面可能删

    request\_queue\_size = 10

    def \_\_init\_\_(self, server\_address):

        # 创建socket，利用socket获取客户端的请求

        self.listen\_socket = socket.socket(

            self.address\_family,

            self.socket\_type

        )

        # 设置socket的工作模式

        self.listen\_socket.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

        # 绑定地址

        self.listen\_socket.bind(server\_address)

        # 激活需求队列

        self.listen\_socket.listen(self.request\_queue\_size)

        # 获得本地服务IP和端口

        # host, port = self.listen\_socket.getsockname()[:2]

        # 根据设置获取IP和端口

        host, port = SERVER\_ADDRESS

        # 获得服务器别名

        self.server\_name = socket.getfqdn(host)

        self.server\_port = port

        # 返回的头信息Return headers set by Web framework/Web application

        self.headers\_set = []

    # 设置server对接的应用程序（处理程序）

    def set\_app(self, application):

        self.application = application

    # 启动WSGI server服务，不停的监听并获取socket数据。

    def serve\_forever(self):

        print('WSGIServer: Serving HTTP on port {port} ...\n'.format(port=PORT))

        while True:  # 无限循环监听

            # 获得监听到的socket请求

            client\_socket, client\_address = self.listen\_socket.accept()

            # 创建新线程

            new\_thread = threading.Thread(target=self.handle\_one\_request, args=(client\_socket,))

            # 启动新线程

            new\_thread.start()

    # 解决请求函数

    def handle\_one\_request(self, client\_socket):

        # 获取请求数据

        self.request\_data = request\_data = client\_socket.recv(1024).decode('utf8')

        if request\_data:

            '''

            # 逐行打印请求报文

            print(''.join(

                '< {line}\n'.format(line=line)

                for line in request\_data.splitlines()

            ))

            '''

            # 分解报文，得到请求报文信息

            # 用需求信息组成环境字典

            env = self.parse\_request(request\_data)

            # 给应用传递两个参数，environ，start\_response

            result = self.application(env, self.start\_response)

            # 用从web应用传回的response给客户发送完成请求报文

            self.finish\_response(result, client\_socket)

        else:

            client\_socket.close()

    # 分解报文

    def parse\_request(self, text):

        # 获得报文第一行，拆分

        request\_line = text.splitlines()[0]

        request\_line = request\_line.rstrip('\r\n')

        # 拆分各个模块信息

        (request\_method,  # GET/POST

         path,  # '/...'

         request\_version  # HTTP/1.1

         ) = request\_line.split()

        return self.get\_environ(request\_method, path)

    # 获取environ数据并设置当前server的工作模式

    def get\_environ(self, request\_method, path):

        # werkzeug\_request = Request()

        env = {

            'wsgi.version': (1, 0),

            'wsgi.url\_scheme': 'http',

            'wsgi.input': io.StringIO(self.request\_data),

            'wsgi.errors': sys.stderr,

            'wsgi.multithread': True,

            'wsgi.multiprocess': False,

            'wsgi.run\_once': False,

            'REQUEST\_METHOD': request\_method,

            'PATH\_INFO': path,

            'SERVER\_NAME': self.server\_name,

            'SERVER\_PORT': str(self.server\_port)

        }

        # 返回环境值

        return env

    # 初始化返回报文

    def start\_response(self, status, response\_headers, exc\_info=None):

        # 必要的报文要素，记录报文发送时间和server版本

        now\_time = datetime.datetime.now()

        str\_now\_time = datetime.datetime.strftime(now\_time, '%Y-%m-%d %H:%M:%S')

        server\_headers = [

            ('Date', str\_now\_time),

            ('Server', 'WSGIServer 0.2'),

        ]

        self.headers\_set = [status, response\_headers + server\_headers]

    # 完成回复报文

    def finish\_response(self, result, client\_socket):

        try:

            # 获取返回报文头信息

            status, response\_headers = self.headers\_set

            # 准备发送报文头

            response\_head = 'HTTP/1.1 {status}\r\n'.format(status=status)

            response = 'HTTP/1.1 {status}\r\n'.format(status=status)

            for header in response\_headers:

                response\_head += '{0}: {1}\r\n'.format(\*header)

                response += '{0}: {1}\r\n'.format(\*header)

            response\_head += '\r\n'

            response += '\r\n'

            # 准备发送报文体

            response\_body = b''

            for data in result:

                response += str(data, encoding="utf-8")

                response\_body += data

            '''

            # 输出报文内容

            print(''.join(

                '> {line}\n'.format(line=line)

                for line in response.splitlines()

            ))

            '''

            client\_socket.send(response\_head.encode('GBK'))

            client\_socket.send(response\_body)

            # client\_socket.sendall(response.encode('utf8'))

        except Exception:

            print('Exception')

# 创建server实例

def make\_server(server\_address, application):

    server = WSGIServer(server\_address)

    server.set\_app(application)

    return server

if \_\_name\_\_ == '\_\_main\_\_':

    # 基础参数设置

    config = configparser.ConfigParser()

    config.read('config.ini', encoding='utf-8')

    HOST = config.get("ipconfig", "HOST")

    PORT = config.getint("ipconfig", "PORT")

    SERVER\_ADDRESS = (HOST, PORT)

    # web应用调用路径

    app\_path = 'flask\_app:flask\_app'  # sys.argv[1]

    # 分解文件名和app实例名

    \_module, \_application = app\_path.split(':')

    # import需求文件

    module = \_\_import\_\_(\_module)

    \_application = getattr(module, \_application)

    # 创建server

    httpd = make\_server(SERVER\_ADDRESS, \_application)

    # 一直运行server

    httpd.serve\_forever()

**数据库创建源代码（BasicDB.sql）**

drop database if exists bank\_db;

create database if not exists bank\_db;

use bank\_db;

drop table if exists web\_account;

create table if not exists web\_account(

    username varchar(10) primary key comment'web账户',

    web\_password varchar(15) not null comment'web密码',

    acc\_power boolean not null comment'账户权限',

    id\_num char(18) comment'身份证号',

    client\_location varchar(50) comment'地址',

    telephone char(11) comment'手机号',

    client\_name varchar(10) comment'开户姓名',

    web\_status boolean not null comment'web账户状态',

    connect\_status boolean not null comment'连接状态'

)charset=utf8 comment='web账户表';

drop table if exists web\_associ\_bank;

create table if not exists web\_associ\_bank(

    username varchar(10) primary key comment'web账户',

    bank\_card\_id char(5) not null comment'关联web账户名',

    associ\_date datetime not null comment'关联时间',

    associ\_status boolean not null comment'关联状态'

)charset=utf8 comment='web\_bank关联表';

drop table if exists bank\_account;

create table if not exists bank\_account(

    bank\_card\_id char(5) primary key comment'银行卡卡号',

    card\_password varchar(15) not null comment'银行卡密码',

    acc\_start\_date datetime not null comment'开户时间',

    acc\_end\_date datetime comment'销户时间',

    left\_money decimal(9,2) not null comment'余额',

    bank\_status boolean not null comment'账户状态'

)charset=utf8 comment='银行卡表';

drop table if exists web\_acc\_log;

create table if not exists web\_acc\_log(

    log\_record\_id int(20) primary key comment'登录记录唯一编号',

    username varchar(10) not null comment'登录者',

    log\_ip varchar(20) not null comment'登录ip地址'

)charset=utf8 comment='登录记录关联表';

drop table if exists acc\_log\_record;

create table if not exists acc\_log\_record(

    log\_record\_id int(20) primary key comment'账户操作记录唯一编号',

    log\_oper\_time datetime comment'账户操作时间',

    log\_kind varchar(15) comment'账户操作类型'

)charset=utf8 comment='账户登录记录表';

drop table if exists acc\_oper\_record;

create table if not exists acc\_oper\_record(

    oper\_record\_id int(20) primary key comment'操作记录唯一编号',

    oper\_date datetime comment'操作时间',

    oper\_kind varchar(10) not null comment'操作类型',

    do\_oper varchar(10) not null comment'操作发出者',

    fin\_oper varchar(10) comment'被操作者',

    oper\_value decimal(9,2) comment'操作值',

    oper\_result boolean not null comment'操作结果',

    remark text comment'备注'

)charset=utf8 comment='对银行卡内容的操作记录表';

drop table if exists bank\_acc\_oper;

create table if not exists bank\_acc\_oper(

    oper\_record\_id int(20) primary key auto\_increment comment'操作记录唯一编号',

    bank\_card\_id char(19) comment'银行卡卡号'

)charset=utf8 comment='银行卡操作记录关联表';

set global log\_bin\_trust\_function\_creators=TRUE;

set @x=1;

delimiter $$

drop function if exists getmoney$$

create function getmoney(card\_id char(5)) returns decimal(9,2)

begin

    declare return\_money decimal(9,2);

    select left\_money from bank\_account where bank\_card\_id=card\_id into return\_money;

    return return\_money;

end $$

delimiter ;

delimiter $$

drop procedure if exists create\_new\_web\_acc$$

create procedure create\_new\_web\_acc(in username varchar(10),in web\_password varchar(15),in id\_num char(18),in client\_location varchar(50),in telephone char(11),in client\_name varchar(10))

begin

    insert into web\_account values(

        username,

        web\_password,

        false,

        id\_num,

        client\_location,

        telephone,

        client\_name,

        true,

        false

    );

end $$

delimiter ;

delimiter $$

drop procedure if exists insert\_log\_record$$

create procedure insert\_log\_record(in temp\_log\_id int(8),in log\_kind varchar(15),in username varchar(10),in log\_ip varchar(20))

begin

    insert into acc\_log\_record values(temp\_log\_id,now(),log\_kind);

    insert into web\_acc\_log values(temp\_log\_id,username,log\_ip);

end $$

delimiter ;

delimiter $$

drop procedure if exists insert\_oper\_record$$

create procedure insert\_oper\_record(in temp\_oper\_id int(8),in oper\_kind varchar(10),in do\_oper varchar(10),in fin\_oper varchar(10),in oper\_value decimal(9,2),in oper\_result boolean,in remark text)

begin

    insert into acc\_oper\_record values(temp\_oper\_id,now(),oper\_kind,do\_oper,fin\_oper,oper\_value,oper\_result,remark);

    insert into bank\_acc\_oper values(temp\_oper\_id,bank\_card\_id);

end $$

delimiter ;

delimiter $$

drop procedure if exists delete\_web\_acc$$

create procedure delete\_web\_acc(in id int(8),in delusername varchar(10),in ip int(10))

begin

    call deconnect\_web\_bank(delusername);

    call insert\_log\_record(id,'deleteaccount',delusername,ip);

    update web\_account set web\_status=false where username=delusername;

end $$

delimiter ;

delimiter $$

drop procedure if exists logout\_acc$$

create procedure logout\_acc(in id int(8),in username varchar(10),in ip int(10))

begin

    call insert\_log\_record(id,'logout',username,ip);

end $$

delimiter ;

delimiter $$

drop procedure if exists check\_power$$

create procedure check\_power(in username varchar(10),in temp\_password varchar(15))

begin

    declare acc\_num int;

    declare get\_power boolean;

    select count(\*) from web\_account where username=username and web\_password=temp\_password into acc\_num;

    if acc\_num=1 then

    select acc\_power from web\_account where username=username into get\_power;

    else

    set get\_power=null;

    end if;

    select get\_power;

end $$

delimiter ;

delimiter $$

drop procedure if exists check\_bank\_acc$$

create procedure check\_bank\_acc(in card\_id char(5),in card\_pass varchar(15),out temp\_status boolean)

begin

    declare acc\_count int;

    select count(\*) from bank\_account where bank\_card\_id=card\_id and card\_password=card\_pass into acc\_count;

    if acc\_count=1 then

    select bank\_status from bank\_account where bank\_card\_id=card\_id into temp\_status;

    else

    set temp\_status=null;

    end if;

end $$

delimiter ;

delimiter $$

drop procedure if exists connect\_web\_bank$$

create procedure connect\_web\_bank(in username varchar(10),in card\_id char(5),in card\_pass varchar(15),out con\_status boolean)

begin

    declare bk\_status boolean default false;

    call check\_bank\_acc(card\_id,card\_pass,bk\_status);

    case bk\_status

    when false then

    update web\_account set connect\_status=true where username=username;

    update bank\_account set bank\_status=true where bank\_card\_id=card\_id;

    insert into web\_associ\_bank values(username,card\_id,now(),true);

    set con\_status=true;

    else

    set con\_status=false;

    end case;

end $$

delimiter ;

delimiter $$

drop procedure if exists deconnect\_web\_bank$$

create procedure deconnect\_web\_bank(in id int(8),in delusername varchar(10),out decon\_status boolean)

begin

    declare card\_id char(5);

    select bank\_card\_id from web\_associ\_bank where username=delusername and associ\_status=true into card\_id;

    call insert\_oper\_record(id,'deconnect',card\_id,card\_id,0.00,true,null);

    update bank\_account set bank\_status=false where bank\_card\_id=card\_id;

    update web\_account set connect\_status=false where username=delusername;

    delete from web\_associ\_bank where username=delusername;

    set decon\_status=true;

end $$

delimiter ;

delimiter $$

drop procedure if exists select\_log\_record$$

create procedure select\_log\_record(in username varchar(10))

begin

    select log\_oper\_time,log\_kind,log\_ip from web\_acc\_log natural join acc\_log\_record where username=username;

end $$

delimiter ;

delimiter $$

drop procedure if exists select\_oper\_record$$

create procedure select\_oper\_record(in id int(8),in card\_id char(5))

begin

    call insert\_oper\_record(id,'check',card\_id,card\_id,0.00,true,null);

    select oper\_date,oper\_kind,do\_oper,fin\_oper,oper\_value,oper\_result,remark from acc\_oper\_record natural join bank\_acc\_oper  where bank\_card\_id=card\_id;

end $$

delimiter ;

delimiter $$

drop procedure if exists deposit$$

create procedure deposit(in id int(8),in card\_id char(5),in add\_value decimal(9,2))

begin

    declare fin\_st boolean;

    call insert\_oper\_record(id,'deposit',card\_id,card\_id,add\_value,true,null);

    update bank\_account set left\_money=left\_money+add\_value where bank\_card\_id=card\_id;

    set fin\_st=true;

    select fin\_st;

end $$

delimiter ;

delimiter $$

drop procedure if exists withdrawal$$

create procedure withdrawal(in id int(8),in card\_id char(5),in dec\_value decimal(9,2))

begin

    declare left\_mon decimal(9,2);

    declare fin\_st boolean;

    set left\_mon=getmoney(card\_id);

    set left\_mon=left\_mon-dec\_value;

    if left\_mon>0 then

    call insert\_oper\_record(id,'withdrawal',card\_id,card\_id,dec\_value,true,null);

    update bank\_account set left\_money=left\_money-dec\_value where bank\_card\_id=card\_id;

    set fin\_st=true;

    else

    call insert\_oper\_record(id,'withdrawal',card\_id,card\_id,dec\_value,flase,null);

    set fin\_st=false;

    end if;

    select fin\_st;

end $$

delimiter ;

delimiter $$

drop procedure if exists transfer\_money$$

create procedure transfer\_money(in id int(8),in do\_card\_id char(5),in to\_card\_id char(5),in trans\_value decimal(9,2))

begin

    declare left\_mon decimal(9,2);

    declare fin\_st boolean;

    set left\_mon=getmoney(do\_card\_id);

    set left\_mon=left\_mon-trans\_value;

    if left\_mon>0 then

    call insert\_oper\_record(id,'transfer',do\_card\_id,to\_card\_id,trans\_value,true,null);

    update bank\_account set left\_money=left\_money-trans\_value where bank\_card\_id=do\_card\_id;

    update bank\_account set left\_money=left\_money+trans\_value where bank\_card\_id=to\_card\_id;

    set fin\_st=true;

    else

    call insert\_oper\_record(id,'transfer',do\_card\_id,to\_card\_id,trans\_value,flase,null);

    set fin\_st=false;

    end if;

    select fin\_st;

end $$

delimiter ;

insert into web\_account

values('abcd','16415636867',false,'130421198011154478','Beijing','13642345112','Zhangsan',true,false),

      ('efg','45984848449',false,'350402199401267511','Shandong','13318877954','Lisi',true,false),

      ('hyjk','14587716458',true,'211381198301121510','Hebei','13642345112','Wangwu',true,false);

insert into bank\_account values('16516','16415636867',now(),null,500.00,false),

    ('91565','45984848449',now(),null,900.00,false);

insert into acc\_oper\_record

values(101472,now(),'deposit',91565,91565,5959.00,1,''),

(109472,now(),'deposit',91565,91565,2000.00,1,''),

(117270,now(),'deposit',91565,91565,800.00,1,''),

(125083,now(),'deposit',91565,91565,9000.00,1,''),

(252920,now(),'deposit',91565,91565,6000.00,1,''),

(252925,now(),'deposit',91565,91565,400.00,1,''),

(292920,now(),'deposit',91565,91565,900.00,1,'');

insert into acc\_oper\_record

values(101372,now(),'deposit',16516,16516,5959.00,1,''),

(101672,now(),'deposit',16516,16516,2000.00,1,''),

(117870,now(),'deposit',16516,16516,800.00,1,''),

(122083,now(),'deposit',16516,16516,9000.00,1,''),

(252420,now(),'deposit',16516,16516,6000.00,1,''),

(252965,now(),'deposit',16516,16516,400.00,1,''),

(292950,now(),'deposit',16516,16516,900.00,1,'');

drop user if exists 'client'@'localhost';

drop user if exists 'administor'@'localhost';

create user if not exists 'client'@'localhost' identified by '123456789';

create user if not exists'administor'@'localhost' identified by '20001227';

grant execute on function getmoney to 'client'@'localhost';

grant execute on procedure create\_new\_web\_acc to 'client'@'localhost';

grant execute on procedure insert\_log\_record to 'client'@'localhost';

grant execute on procedure insert\_oper\_record to 'client'@'localhost';

grant execute on procedure delete\_web\_acc to 'client'@'localhost';

grant execute on procedure logout\_acc to 'client'@'localhost';

grant execute on procedure check\_power to 'client'@'localhost';

grant execute on procedure check\_bank\_acc to 'client'@'localhost';

grant execute on procedure connect\_web\_bank to 'client'@'localhost';

grant execute on procedure deconnect\_web\_bank to 'client'@'localhost';

grant execute on procedure select\_log\_record to 'client'@'localhost';

grant execute on procedure select\_oper\_record to 'client'@'localhost';

grant execute on procedure deposit to 'client'@'localhost';

grant execute on procedure withdrawal to 'client'@'localhost';

grant execute on procedure transfer\_money to 'client'@'localhost';

grant execute on function getmoney to 'administor'@'localhost';

grant execute on procedure create\_new\_web\_acc to 'administor'@'localhost';

grant execute on procedure insert\_log\_record to 'administor'@'localhost';

grant execute on procedure insert\_oper\_record to 'administor'@'localhost';

grant execute on procedure delete\_web\_acc to 'administor'@'localhost';

grant execute on procedure logout\_acc to 'administor'@'localhost';

grant execute on procedure check\_power to 'administor'@'localhost';

grant execute on procedure check\_bank\_acc to 'administor'@'localhost';

grant execute on procedure connect\_web\_bank to 'administor'@'localhost';

grant execute on procedure deconnect\_web\_bank to 'administor'@'localhost';

grant execute on procedure select\_log\_record to 'administor'@'localhost';

grant execute on procedure select\_oper\_record to 'administor'@'localhost';

grant execute on procedure deposit to 'administor'@'localhost';

grant execute on procedure withdrawal to 'administor'@'localhost';

grant execute on procedure transfer\_money to 'administor'@'localhost';

grant select on bank\_db.\* to 'administor'@'localhost';

**前端代码（HTML文件）太多，放在templates文件夹中**

1. **附件二：视频（PPT）（如果有）**