四川大學

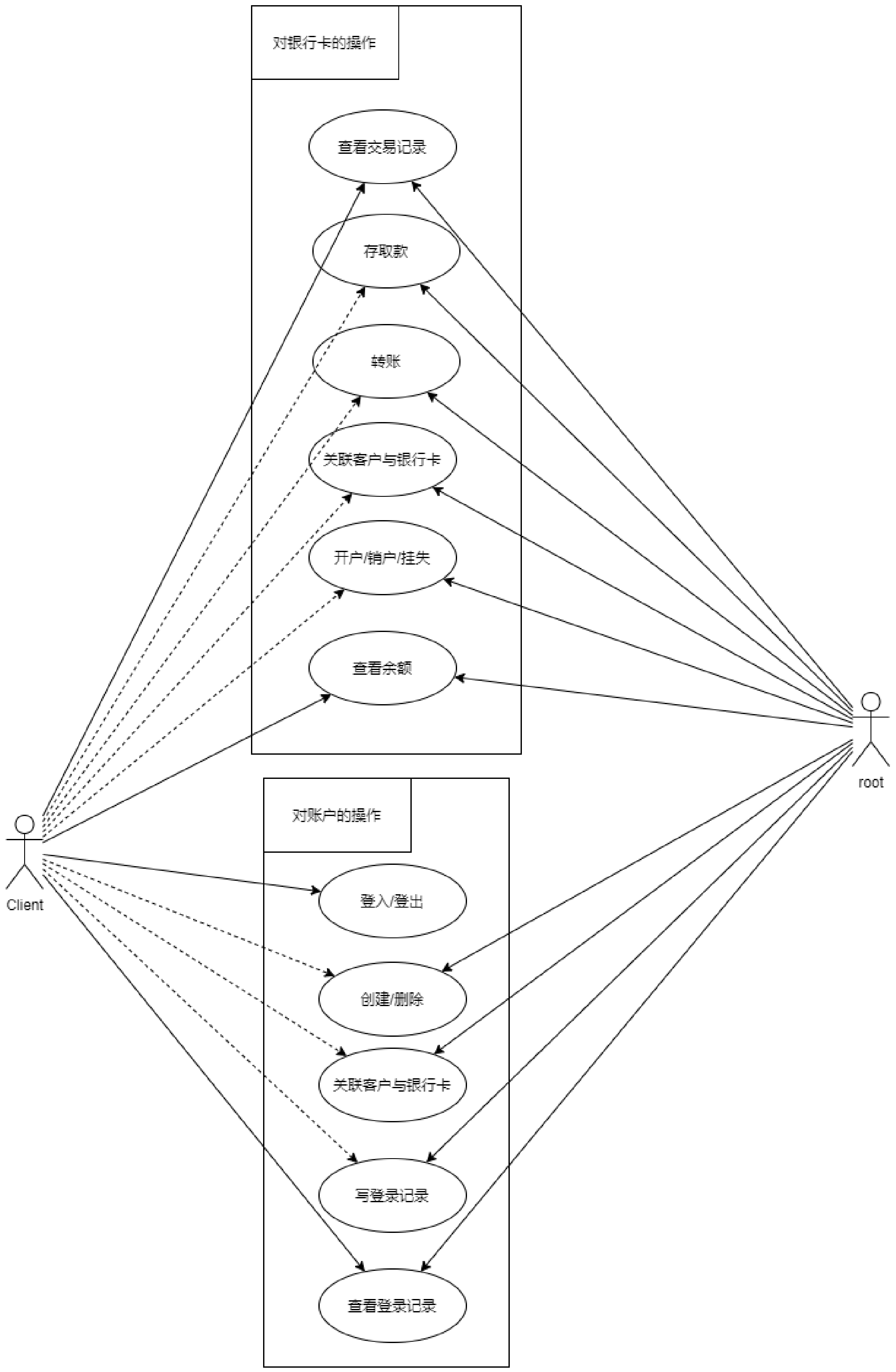
计算机网络
项目开发报告

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

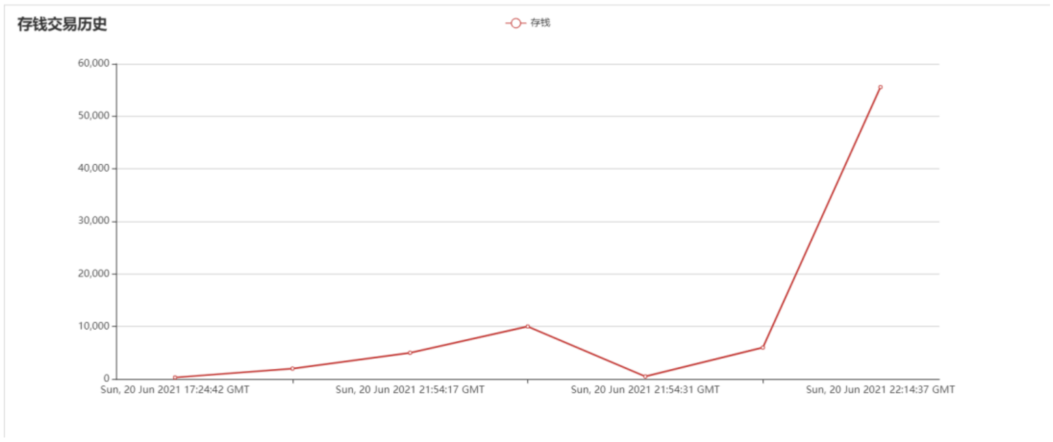
一、 功能及说明



（一）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。

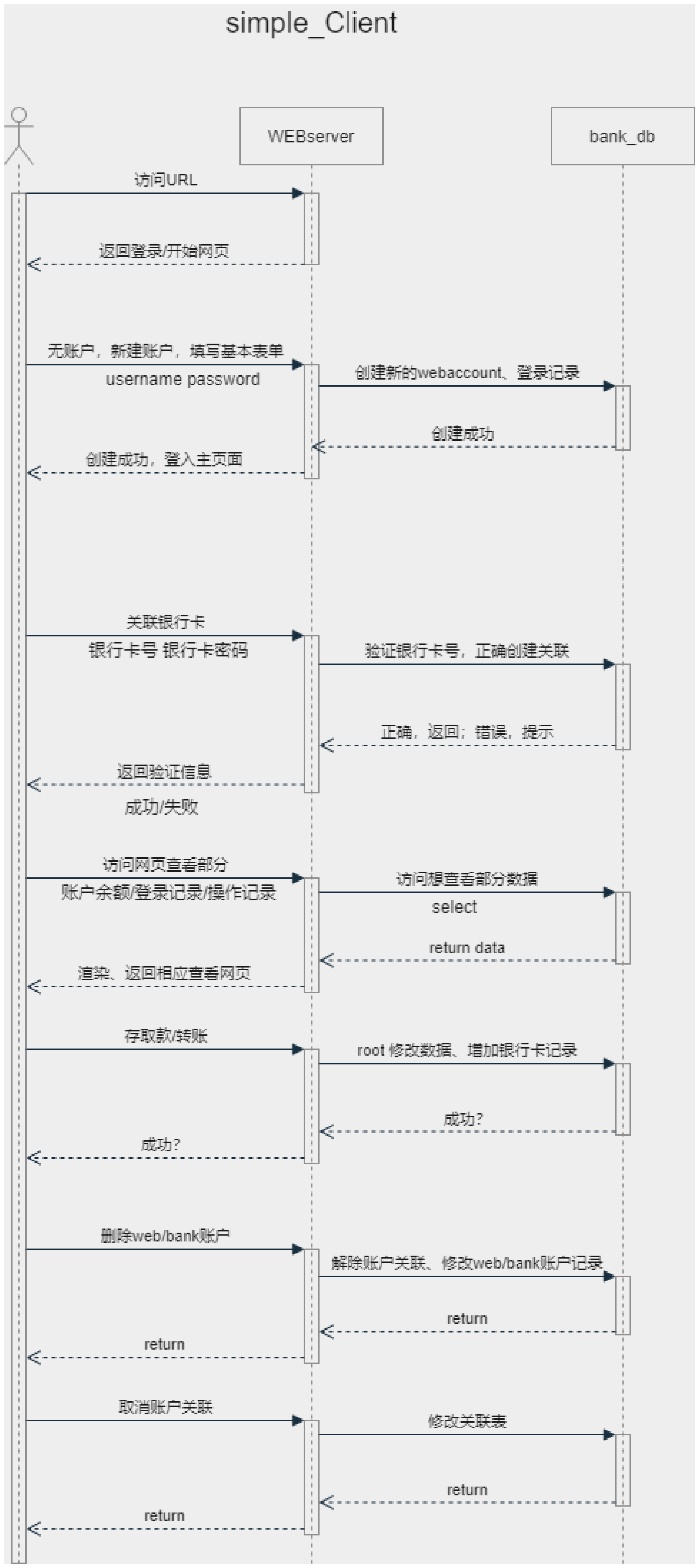
（二）建立银行数据库系统建立用户信息，支持不同权限数据库用户增、删、改、查等操作。

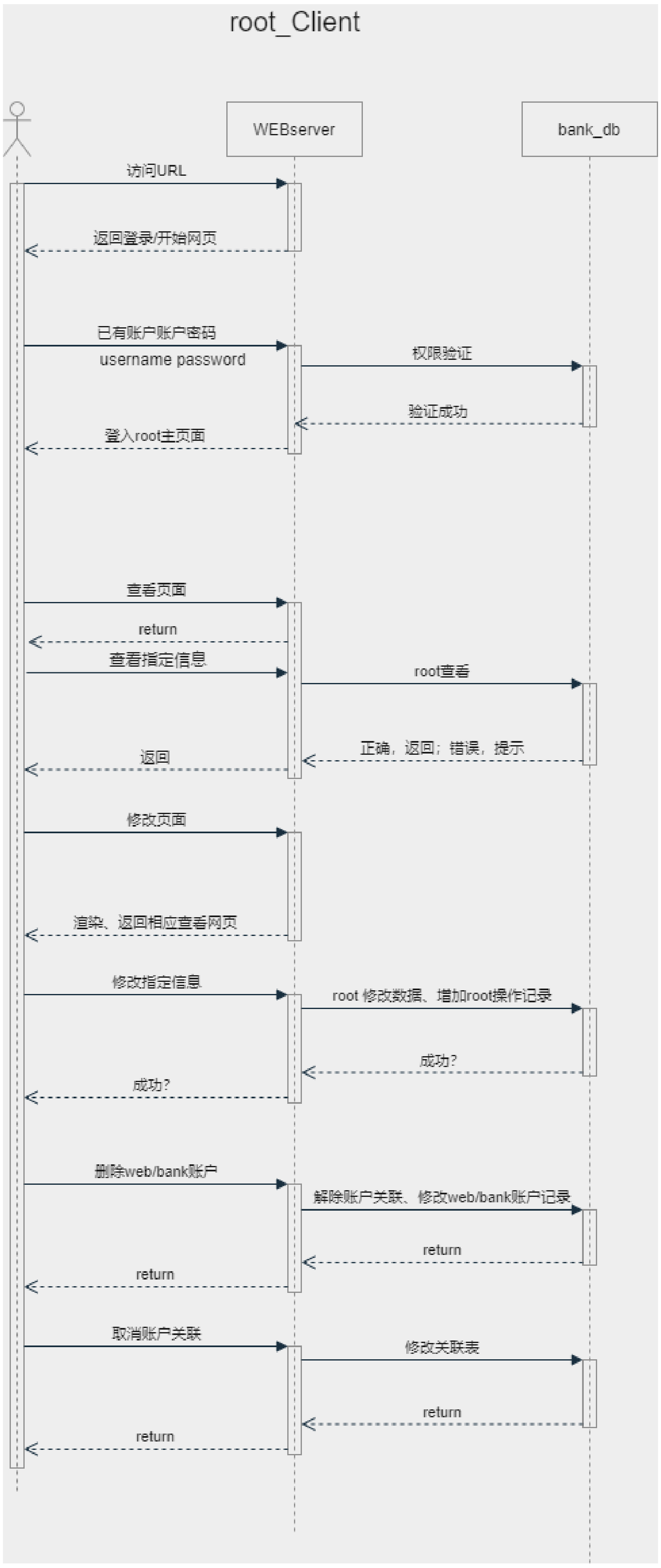
（三）用户可以通过远程访问银行 web 网页完成对账户的查询、转账等功能的操作



（四）数据库管理员可以有不同操作权限

二、 实现功能的流程及说明


simple_Client
WEBserver
bank_db
访问URL
返回登录/开始网页
无账户，新建账户，填写基本表单
username password
创建新的webaccount、登录记录
创建成功
创建成功，登入主页面
关联银行卡
银行卡号 银行卡密码
验证银行卡号，正确创建关联
正确，返回；错误，提示
返回验证信息
成功/失败
访问网页查看部分
账户余额/登录记录/操作记录
访问想查看部分数据
select
return data
渲染、返回相应查看网页
存取款/转账
root 修改数据、增加银行卡记录
成功？
成功？
删除web/bank账户
解除账户关联、修改web/bank账户记录
return
return
取消账户关联
修改关联表
return
return


root_Client
WEBserver
bank_db
访问URL
返回登录/开始网页
已有账户账户密码
username password
权限验证
验证成功
登入root主页面
查看页面
return
查看指定信息
root查看
正确，返回；错误，提示
返回
修改页面
渲染、返回相应查看网页
修改指定信息
root 修改数据、增加root操作记录
成功？
成功？
删除web/bank账户
解除账户关联、修改web/bank账户记录
return
return
取消账户关联
修改关联表
return
return

（一） 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。

1. 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。
2. 通过python多线程技术允许同时多用户访问网页。

（二） 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。

1. 通过数据库设计，事先建立服务器本地数据库。
2. 通过管理员导入、修改用户信息维护服务器数据库。

（三） 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

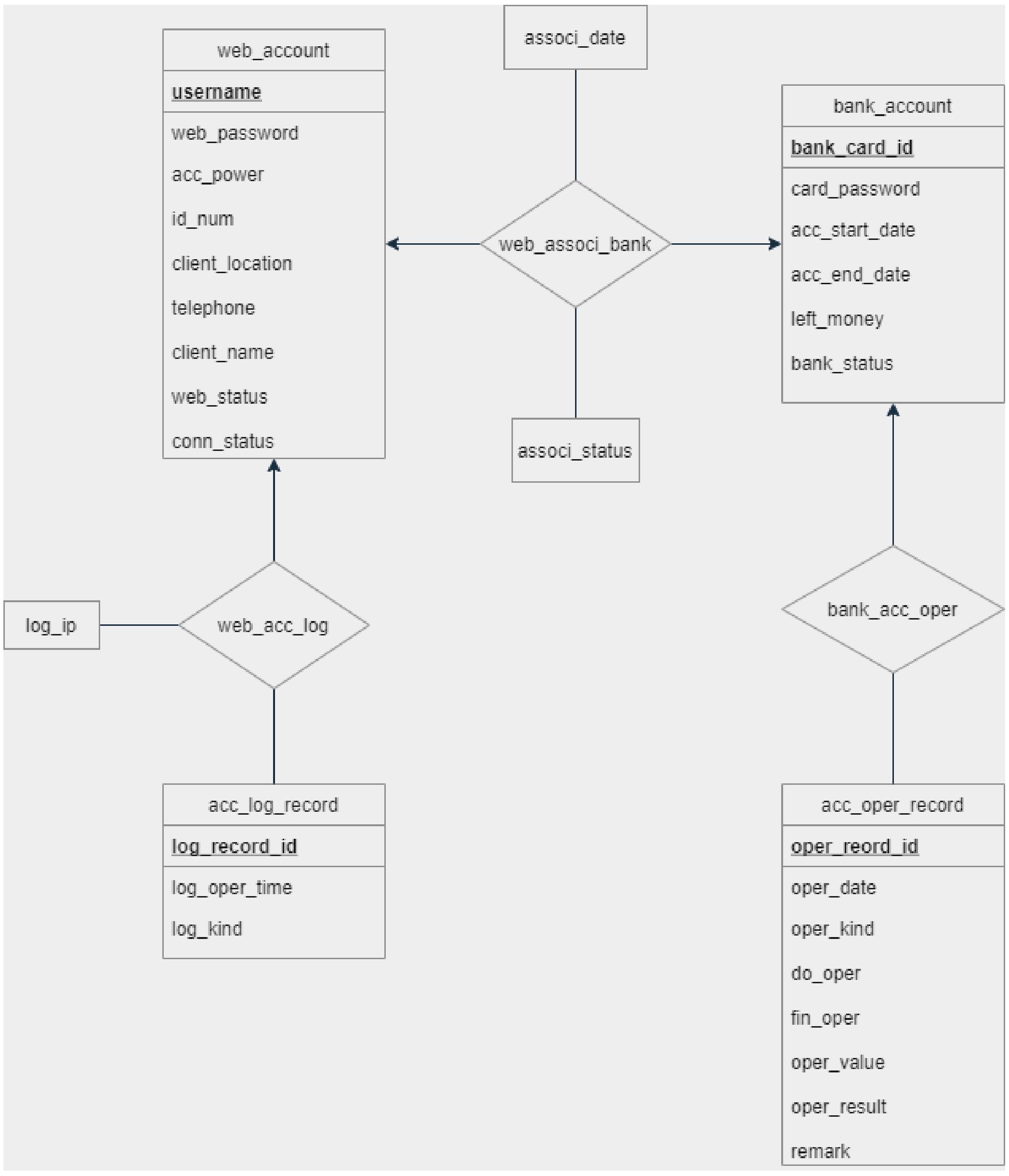
1. 用户通过账户、密码登录进入个人页操作。
2. 进入个人页，通过选项（关联银行卡、查询账户信息变更流水记录、转账、存入……）。
3. 退出网页。

（四） 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

1. 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
2. 管理员退出登录

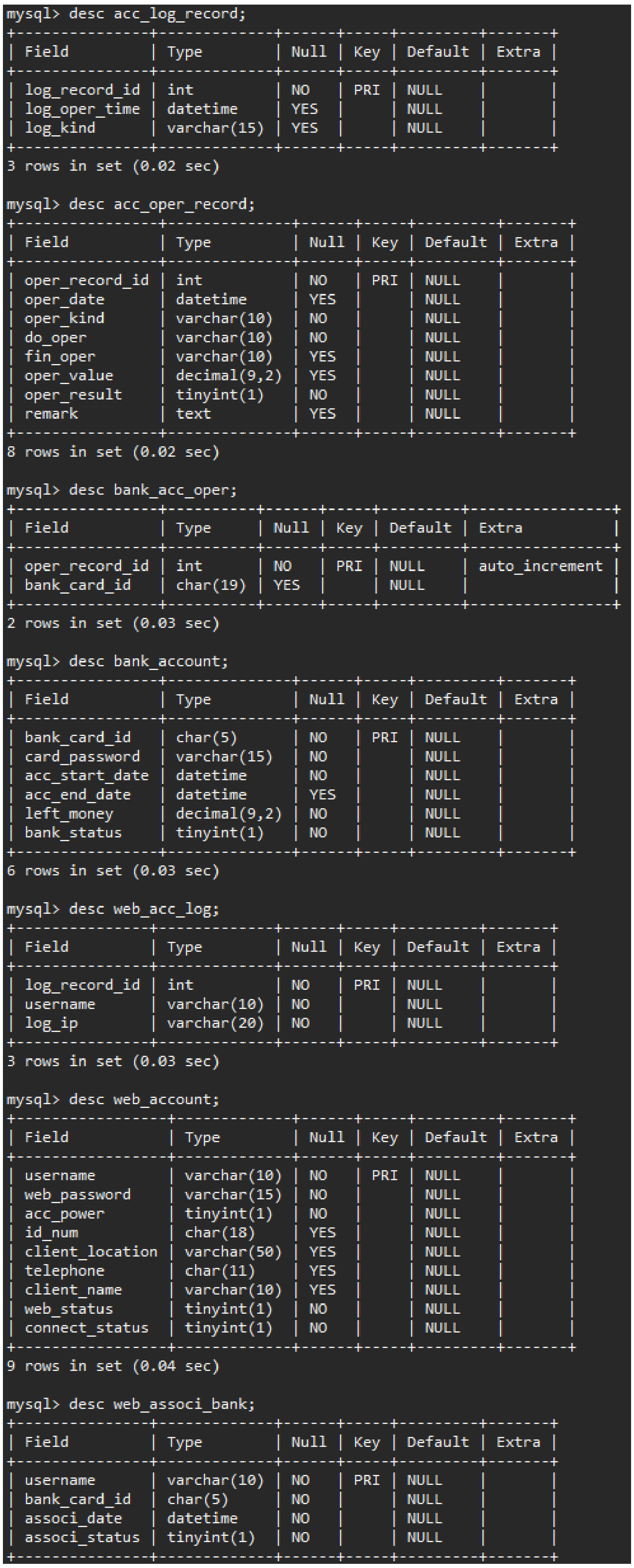
三、 数据库设计与说明

E-R图：

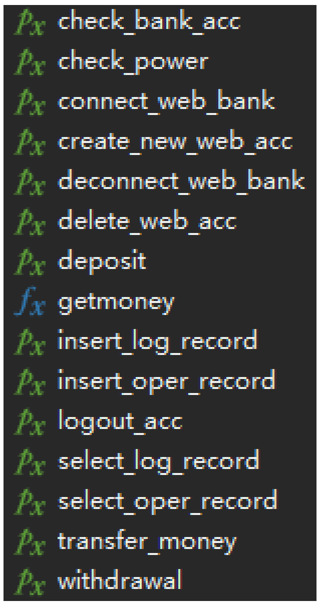


一共七个关系，即 E-R 图中所示，实体有四个，分别是网页账户（web_account）、银行卡账户（bank_account）、网页登录登出记录（acc_log_record）、银行卡账户操作记录（acc_oper_record）网页账户与银行卡账户可能存在一一关联的关系，取决于是否绑定银行卡，账户的登录、对账户的操作记录在登入登出的时候自动记录，在关联表（web_acc_log、bank_acc_oper）中将记录与账户对应。

所有表的字段格式如下



```
mysql> desc acc_log_record;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| log_record_id | int         | NO   | PRI | NULL    |       |
| log_oper_time | datetime    | YES  |     | NULL    |       |
| log_kind      | varchar(15) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
3 rows in set (0.02 sec)

mysql> desc acc_oper_record;
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
| oper_record_id | int          | NO   | PRI | NULL    |       |
| oper_date      | datetime     | YES  |     | NULL    |       |
| oper_kind      | varchar(10)  | NO   |     | NULL    |       |
| do_oper        | varchar(10)  | NO   |     | NULL    |       |
| fin_oper       | varchar(10)  | YES  |     | NULL    |       |
| oper_value     | decimal(9,2) | YES  |     | NULL    |       |
| oper_result    | tinyint(1)   | NO   |     | NULL    |       |
| remark         | text         | YES  |     | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
8 rows in set (0.02 sec)

mysql> desc bank_acc_oper;
+----------------+----------+------+-----+---------+----------------+
| Field          | Type     | Null | Key | Default | Extra          |
+----------------+----------+------+-----+---------+----------------+
| oper_record_id | int      | NO   | PRI | NULL    | auto_increment |
| bank_card_id   | char(19) | YES  |     | NULL    |                |
+----------------+----------+------+-----+---------+----------------+
2 rows in set (0.03 sec)

mysql> desc bank_account;
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
| bank_card_id   | char(5)      | NO   | PRI | NULL    |       |
| card_password  | varchar(15)  | NO   |     | NULL    |       |
| acc_start_date | datetime     | NO   |     | NULL    |       |
| acc_end_date   | datetime     | YES  |     | NULL    |       |
| left_money     | decimal(9,2) | NO   |     | NULL    |       |
| bank_status    | tinyint(1)   | NO   |     | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
6 rows in set (0.03 sec)

mysql> desc web_acc_log;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| log_record_id | int         | NO   | PRI | NULL    |       |
| username      | varchar(10) | NO   |     | NULL    |       |
| log_ip        | varchar(20) | NO   |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
3 rows in set (0.03 sec)

mysql> desc web_account;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| username        | varchar(10) | NO   | PRI | NULL    |       |
| web_password    | varchar(15) | NO   |     | NULL    |       |
| acc_power       | tinyint(1)  | NO   |     | NULL    |       |
| id_num          | char(18)    | YES  |     | NULL    |       |
| client_location | varchar(50) | YES  |     | NULL    |       |
| telephone       | char(11)    | YES  |     | NULL    |       |
| client_name     | varchar(10) | YES  |     | NULL    |       |
| web_status      | tinyint(1)  | NO   |     | NULL    |       |
| connect_status  | tinyint(1)  | NO   |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
9 rows in set (0.04 sec)

mysql> desc web_associ_bank;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| username      | varchar(10) | NO   | PRI | NULL    |       |
| bank_card_id  | char(5)     | NO   |     | NULL    |       |
| associ_date   | datetime    | NO   |     | NULL    |       |
| associ_status | tinyint(1)  | NO   |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
```


check_bank_acc
check_power
connect_web_bank
create_new_web_acc
deconnect_web_bank
delete_web_acc
deposit
getmoney
insert_log_record
insert_oper_record
logout_acc
select_log_record
select_oper_record
transfer_money
withdrawal

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

四、 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

五、 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，将用户信息掺入其中

六、 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;
```

```sql
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_pass
word 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_st
atus 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_pa
ss 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_s
tatus boolean)
begin
    declare card_id char(5);
    select bank_card_id from web_associ_bank where username=delusername and associ_stat
us=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 w
here 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';
```

```sql
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

七、 附件二：视频（PPT）（如果有）