“银行业务管理系统”

系统设计与实现报告

姓名：沈欣怡

学号：**PB17111562**

计算机科学与技术学院

中国科学技术大学

2020 年 4 月

目录

1 概述

1.1 系统目标

设计并实现一个银行业务管理系统。

1.2 需求说明

客户管理：提供客户所有信息的增、删、改、查功能； 如果客户存在着关联账户或者贷款记录，则不允许删除；

账户管理：提供账户开户、销户、修改、查询功能，包括储蓄账户和支票账户； 账户号不允许修改；

贷款管理：提供贷款信息的增、删、查功能，提供贷款发放功能； 贷款信息一旦添加成功后不允许修改； 要求能查询每笔贷款的当前状态（未开始发放、发放中、已全部发放）；处于发放中状态的贷款记录不允许删除；

业务统计： 按业务分类和时间统计各个支行的业务总金额和用户数，对统计结果同时提供表格和曲线图两种可视化展示方式。

1.3 本报告的主要贡献

介绍系统各模块的结构、逻辑和实现，展示实验成果。

2 总体设计

2.1 系统模块结构

采用助教提供的 python+flask 模板，在此基础上完成应用；图标显示使用了 echarts.

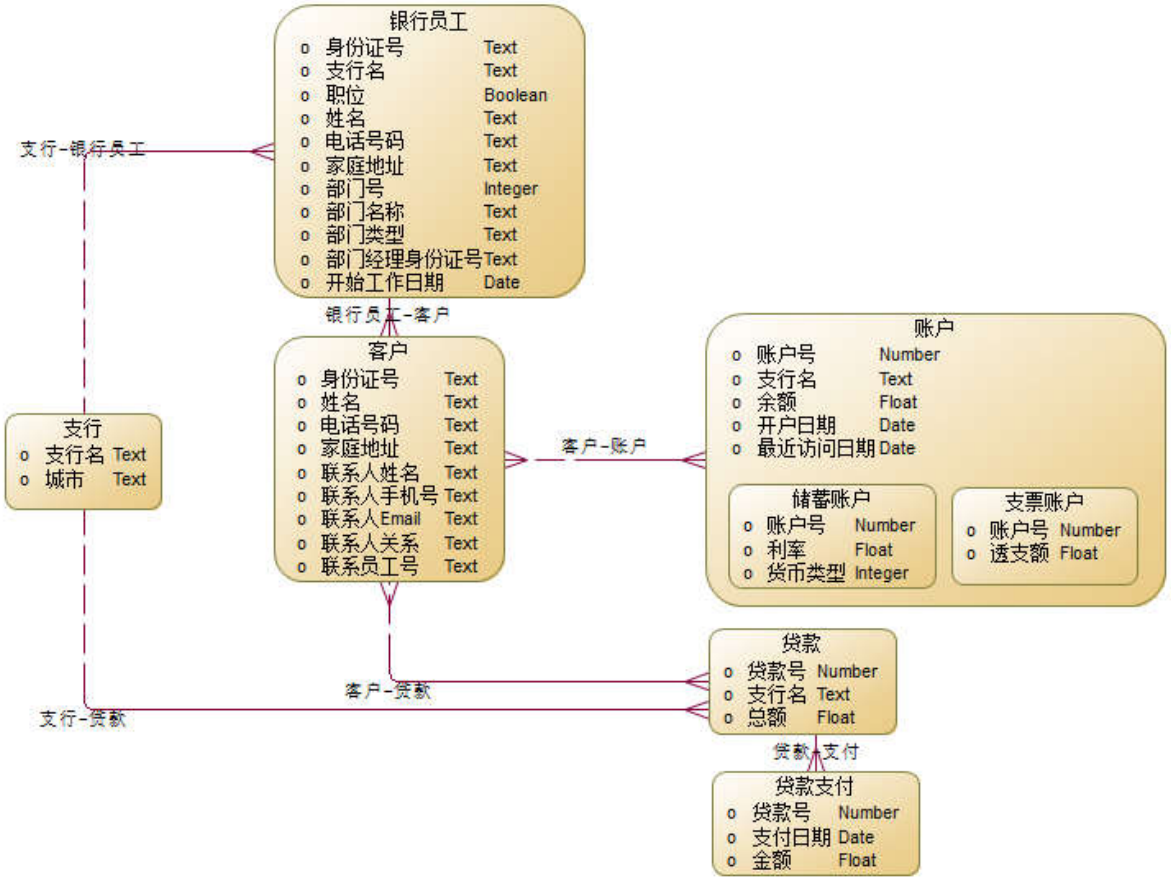
main.py：负责所有页面的输入处理、跳转、数据的读写和传输；

db.py：处理登录、关闭数据库、从数据库读主页需要的数据。

2.2 系统工作流程

页面发出请求—对应函数处理请求—从数据库读数据—处理数据—返回（重链接到新的页面或渲染模板）。对于数据库的管理操作，如果操作成功则回到主页，失败则留在该页并有弹窗提示。

2.3 数据库设计



逻辑与物理结构：

实体：

支行（支行名，城市）

Banks(bank_name, city)

银行员工（身份证号，支行名，职位，姓名，电话号码，家庭住址，部门号，部门名称，部门类型，部门经理身份证号，开始工作日期）

Assistants(id, bank_name, pos, name, tel, addr, sec_num, sec_name, sec_type, sec_mng_id, work_date)

客户（身份证号，电话号码，家庭住址，联系人姓名，联系人手机号，联系人 email，联系人关系，联系员工号）

Clients(id, name, tel, addr, con_name, con_tel, con_email, con_relat, assist_id)

账户（账户号，支行名，余额，开户日期，最近访问日期）
Accounts(acc_num, bank_name, acc_type, balance, open_date, visit_date)
账户的两个子类：
储蓄账户（账户号，利率，货币类型）
Save_accounts(acc_num, int_rate, cur_type)
支票账户（账户号，透支额）
Check_accounts(acc_num, overdraft)
贷款（贷款号，支行名，总额）
Debts(debt_num, bank_name, amount)
贷款支付（贷款号，支付日期，金额）
Debts_pay(debt_num, pay_date, pay_sum)

关系：
客户-账户（客户身份证号，账户号）
Cli_Acc(cli_id, acc_num)
客户-支行-账户类型（客户身份证号，支行名，账户号，账户类型）
Cli_Bank_Acctype(cli_id, bank_name, acc_num, acc_type)
*注：这里的支行名为冗余属性，为了保证一个客户在一个支行内只能开设一个储蓄账户和一个支票账户。账户号也为冗余属性，为了在删除/修改账户时便于查找。
客户-贷款（客户身份证号，贷款号）
Cli_Debt(cli_id, debt_num)

3 详细设计

3.1 主页模块

显示两个表：支行信息和银行职员信息。

其余按钮连接到各个模块。

```
@app.route("/table", methods=["GET", "POST"])
def table():
    if 'username' in session:
        db = db_login(session['username'], session['password'],
                      session['ipaddr'], session['database'])
    else:
        return redirect(url_for('login'))

    banks = db_showbanks(db)
    assists = db_showassists(db)
    db_close(db)
    if request.method == "POST":
```

```
        if 'clear' in request.form:
            return render_template("table.html", banks='', assists='')
        elif 'search' in request.form:
            return render_template("table.html", banks=banks, assists=assists)
        elif 'cli_add' in request.form:
            return redirect(url_for("cli_add"))
        elif 'cli_del' in request.form:
            return redirect(url_for("cli_del"))
        elif 'cli_alt' in request.form:
            return redirect(url_for("cli_alt"))
        elif 'cli_search' in request.form:
            return redirect(url_for("cli_search"))
        elif 'acc_open' in request.form:
            return redirect(url_for("acc_open"))
        elif 'acc_close' in request.form:
            return redirect(url_for("acc_close"))
        elif 'acc_alt' in request.form:
            return redirect(url_for("acc_alt"))
        elif 'acc_search' in request.form:
            return redirect(url_for("acc_search"))
        elif 'debt_add' in request.form:
            return redirect(url_for("debt_add"))
        elif 'debt_del' in request.form:
            return redirect(url_for("debt_del"))
        elif 'debt_search' in request.form:
            return redirect(url_for("debt_search"))
        elif 'debt_pay' in request.form:
            return redirect(url_for("debt_pay"))
        elif 'by_types' in request.form:
            return redirect(url_for("by_types"))
        elif 'by_time' in request.form:
            return redirect(url_for("by_time"))
    else:
        return render_template("table.html", banks=banks, assists=assists)
```

3.2 客户管理模块

添加客户：

用双引号引用客户名，避免因为姓名含有单引号的错误；（姓名含有双引号显然

是非法的）。

关键代码：

```
    if request.method == "POST":
        if 'return' in request.form:
            return redirect(url_for('table'))
        elif 'cli_add' in request.form:
            cli_id = request.form["cli_id"]
            name = request.form["cli_name"]
            tel = request.form["cli_tel"]
            addr = request.form["cli_addr"]
            con_name = request.form["con_name"]
            con_tel = request.form["con_tel"]
            con_email = request.form["con_email"]
            con_relat = request.form["con_relat"]
            assist_id = request.form["assist_id"]
            if assist_id != "":
                sql = 'INSERT INTO Clients VALUES("%s", "%s", "%s", "%s", "%s", "%s",
"%s", "%s", "%s")'\
                      % (cli_id, name, tel, addr, con_name, con_tel, con_email,
con_relat, assist_id)
            else:
                sql = 'INSERT INTO Clients VALUES("%s", "%s", "%s", "%s", "%s", "%s",
"%s", "%s", null)'\
                      % (cli_id, name, tel, addr, con_name, con_tel, con_email,
con_relat)
            try:
                cursor.execute(sql)
                db.commit()
                return redirect(url_for('table'))
            except:
                db.rollback()
                return render_template("cli/cli_add_fail.html")

    else:
        return render_template("cli/cli_add.html")
```

删除客户：

关键代码：

```
if request.method == "POST":
```

```
        if 'return' in request.form:
            return redirect(url_for('table'))
        elif 'cli_del' in request.form:
            cli_id = request.form["cli_id"]
            sql = 'DELETE FROM Clients WHERE id = "%s"' % (cli_id)
            try:
                cursor.execute(sql)
                db.commit()
                return redirect(url_for('table'))
            except:
                db.rollback()
                return render_template("cli/cli_del_fail.html")
    else:
        return render_template("cli/cli_del.html")
```

修改客户：

包括修改客户个人信息、修改联系人信息、修改（添加、删除）联系的银行员工。

关键代码：

```
    if request.method == "POST":
        if 'return' in request.form:
            return redirect(url_for('table'))
        else:
            sql = ''
            if 'alt_cli' in request.form:
                cli_id = request.form["cli_id"]
                tel = request.form["cli_tel"]
                addr = request.form["cli_addr"]
                sql = 'UPDATE Clients SET tel = "%s", addr = "%s" \
                      WHERE id = "%s"' % (tel, addr, cli_id)
            elif 'alt_con' in request.form:
                cli_id = request.form["cli_id"]
                name = request.form["con_name"]
                tel = request.form["con_tel"]
                email = request.form["con_email"]
                relat = request.form["con_relat"]
                sql = 'UPDATE Clients  SET con_name = "%s", con_tel = "%s", \
                       con_email = "%s", con_relat = "%s" \
                       WHERE id = "%s"' % (name, tel, email, relat, cli_id)
```

```
        elif 'alt_assist' in request.form:
            cli_id = request.form["cli_id"]
            assist_id = request.form["assist_id"]
            if assist_id != "":
                sql = 'UPDATE Clients SET assist_id = "%s" \
                      WHERE id = "%s"' % (assist_id, cli_id)
            else:
                sql = 'UPDATE Clients SET assist_id = null \
                      WHERE id = "%s"' % (cli_id)
        try:
            cursor.execute(sql)
            db.commit()
            return redirect(url_for('table'))
        except:
            db.rollback()
            return render_template("cli/cli_alt_fail.html")
```

查询客户：

根据客户身份证号或姓名查询特定客户，或查询全部客户。

关键代码：

```
if request.method == "POST":
        if 'return' in request.form:
            return redirect(url_for('table'))
        else:
            sql = ''
            if 'by_id' in request.form:
                cli_id = request.form["cli_id"]
                sql = 'SELECT * FROM Clients WHERE id = "%s"' % (cli_id)
            elif 'by_name' in request.form:
                name = request.form["cli_name"]
                sql = 'SELECT * FROM Clients WHERE name = "%s"' % (name)
            elif 'search_all' in request.form:
                sql = 'SELECT * FROM Clients'
            try:
                cursor.execute(sql)
                rows = cursor.fetchall()
                return render_template('cli/cli_search.html', rows = rows)
            except MySQLdb.Error as e:
                return render_template('cli/cli_search.html', rows = '')
```

3.3 账户管理模块

开户：

账户号由系统自动生成。

添加账户时最多关联 5 个客户，至少关联 1 个客户。随后能在修改账户功能下添加或删除客户。

开户日期由根据系统日期自动生成。

账户类型可以选择，然后需要根据提示输入该类账户对应需要的特定的属性，否则将提示错误，开户失败。

关键代码：

```
if request.method == "POST":
        if 'return' in request.form:
            return redirect(url_for('table'))
        elif 'acc_open' in request.form:
            sql_accounts = ""
            bank_name = '"' + request.form["bank_name"] + '"'
            date = '"' + time.strftime("%Y-%m-%d", time.localtime()) + '"'
            cli_id = ["", "", "", "", ""]
            if request.form["cli_id1"] == "":
                cli_id[0] = None
            else:
                cli_id[0] = '"' + request.form["cli_id1"] + '"'

            ......

            if request.form["cli_id5"] == "":
                cli_id[4] = None
            else:
                cli_id[4] = '"' + request.form["cli_id5"] + '"'

            if request.form["acc_type"] == "save":
                acc_type = '"' + "save" + '"'
                int_rate = request.form["int_rate"]
                cur_type = request.form["cur_type"]
                cur_type = int(cur_type)
```

```
    if int_rate=="":
        return render_template("acc/acc_open_fail.html")
    else:
        int_rate = float(int_rate)
elif request.form["acc_type"] == "check":
    acc_type = '"' + "check" + '"'
    overdraft = request.form["overdraft"]
    if overdraft == "":
        return render_template("acc/acc_open_fail.html")
    else:
        overdraft = float(overdraft)

try:
    sql_accounts = 'INSERT INTO Accounts VALUES(null, %s, 0.0, %s, %s, %s)'\
                % (bank_name, date, date, acc_type)
    print(sql_accounts)
    cursor.execute(sql_accounts)
    acc_num = db.insert_id()
    acc_num = int(acc_num)
    sql_acc_type = ""
    if request.form["acc_type"] == "save":
        sql_acc_type = 'INSERT INTO Save_accounts VALUES(%s, %f, %d)'\
                    % (acc_num, int_rate, cur_type)
    elif request.form["acc_type"] == "check":
        sql_acc_type = 'INSERT INTO Check_accounts VALUES(%s, %f)'\
                    % (acc_num, overdraft)
    cursor.execute(sql_acc_type)
    for i in range(5):
        if cli_id[i] != None:
            sql_cli_acc = 'INSERT INTO Cli_Acc VALUES(%s, %d)'\
                        % (cli_id[i], acc_num)
            print(sql_cli_acc)
            cursor.execute(sql_cli_acc)
            sql_cba = 'INSERT INTO Cli_Bank_Acctype VALUES(%s, %s, %d, %s)'\
                    % (cli_id[i], bank_name, acc_num, acc_type)
            cursor.execute(sql_cba)
    db.commit()
    return redirect(url_for('table'))
except:
    db.rollback()
    return render_template("acc/acc_open_fail.html")
```

销户：

为保证一定的安全性，需要输入账户号和与账户关联的一位客户的身份证号才能销户。

关键代码：

```
if request.method == "POST":
    if 'return' in request.form:
        return redirect(url_for('table'))
    elif 'acc_close' in request.form:
        acc_num = int(request.form["acc_num"])
        cli_id = request.form["cli_id"]
        sql_get = 'SELECT balance, acc_type FROM Accounts, Cli_Acc \
                  WHERE Accounts.acc_num = Cli_Acc.acc_num \
                  AND Accounts.acc_num = %d AND Cli_Acc.cli_id = "%s"'\
                  % (acc_num, cli_id)
        print(sql_get)
        try:
            cursor.execute(sql_get)
            balance, acc_type = cursor.fetchall()[0]
            balance = float(balance)
            if balance != 0:
                return render_template("acc/acc_close_fail.html")
        except:
            return render_template("acc/acc_close_fail.html")
        sql1 = 'DELETE FROM Cli_Bank_Acctype WHERE acc_num = %d' % (acc_num)
        sql2 = 'DELETE FROM Cli_Acc WHERE acc_num = %d' % (acc_num)
        if acc_type == "save":
            sql3 = 'DELETE FROM Save_accounts WHERE acc_num = %d' % (acc_num)
        elif acc_type == "check":
            sql3 = 'DELETE FROM Check_accounts WHERE acc_num = %d' % (acc_num)
        sql4 = 'DELETE FROM Accounts WHERE acc_num = %d' % (acc_num)
        try:
            cursor.execute(sql1)
            cursor.execute(sql2)
            cursor.execute(sql3)
            cursor.execute(sql4)
            db.commit()
            return redirect(url_for('table'))
        except:
```

```
            db.rollback()
            return render_template("acc/acc_close_fail.html")
```

修改账户：

包含存款、取款、修改账户所在支行、添加账户关联客户、删除账户关联客户。

对于储蓄账户，取款时将检查余额是否足够；对支票账户，允许在透支额度内的透支取款。

删除账户关联客户时，不能将账户关联的所有客户都删除。

关键代码：

```
if request.method == "POST":
    if 'return' in request.form:
        return redirect(url_for('table'))
    else:
        sql_list = []
        if 'save' in request.form:
            acc_num = int(request.form["acc_num"])
            amount = float(request.form["amount"])
            if amount < 0:
                return render_template("acc/acc_alt_fail.html")
            sql_get = 'SELECT balance FROM Accounts WHERE acc_num = %d' \
                      % acc_num
            try:
                cursor.execute(sql_get)
                balance = cursor.fetchall()[0][0]
                balance = float(balance)
            except:
                return render_template("acc/acc_alt_fail.html")
            date = time.strftime("%Y-%m-%d", time.localtime())
            sql = 'UPDATE Accounts SET balance = %f, visit_date = "%s" \
                   WHERE acc_num = %d' % (balance+amount, date, acc_num)
            sql_list.append(sql)

        elif 'exploit' in request.form:
            acc_num = int(request.form["acc_num"])
            cli_id = request.form["cli_id"]
            amount = float(request.form["amount"])
            sql_get = 'SELECT balance FROM ( SELECT balance \
                      FROM Accounts , Cli_Acc \
```

```
                WHERE Accounts.acc_num = Cli_Acc.acc_num \
                AND Accounts.acc_num = %d AND Cli_Acc.cli_id = %s) a' \
                % (acc_num, cli_id)
        try:
            cursor.execute(sql_get)
            balance = cursor.fetchall()[0][0]
            balance = float(balance)
        except:
            return render_template("acc/acc_alt_fail.html")
        sql_get = 'SELECT acc_type FROM Accounts \
                WHERE acc_num = %d' % (acc_num)
        cursor.execute(sql_get)
        acc_type = cursor.fetchall()[0][0]
        if acc_type == 'save':
            if balance < amount or amount < 0:
                return render_template("acc/acc_alt_fail.html")
        elif acc_type == 'check':
            sql_get = 'SELECT overdraft FROM Check_accounts \
                    WHERE acc_num = %d' % (acc_num)
            cursor.execute(sql_get)
            overdraft = cursor.fetchall()[0][0]
            overdraft = float(overdraft)
            if overdraft+balance < amount or amount < 0:
                return render_template("acc/acc_alt_fail.html")
        date = time.strftime("%Y-%m-%d", time.localtime())
        sql = 'UPDATE Accounts SET balance = %f, visit_date = "%s" \
              WHERE acc_num = %d' % (balance-amount, date, acc_num)
        sql_list.append(sql)

    elif 'alt_bank' in request.form:
        acc_num = int(request.form["acc_num"])
        bank_name = request.form["bank_name"]
        sql1 = 'UPDATE Cli_Bank_Acctype SET bank_name = "%s" \
             WHERE acc_num = %d' % (bank_name, acc_num)
        sql2 = 'UPDATE Accounts SET bank_name = "%s" \
             WHERE acc_num = %d' % (bank_name, acc_num)
        sql_list.append(sql1)
        sql_list.append(sql2)
    elif 'add_cli' in request.form:
        acc_num = int(request.form["acc_num"])
        cli_id = request.form["cli_id"]
```

```
            sql_get = 'SELECT bank_name, acc_type FROM Accounts \
                      WHERE Accounts.acc_num = %d' % (acc_num)
            print(sql_get)
            try:
                cursor.execute(sql_get)
                bank_name, acc_type = cursor.fetchall()[0]
#                print(bank_name, acc_type)
            except:
                return render_template("acc/acc_alt_fail.html")
            sql1 = 'INSERT INTO Cli_Bank_Acctype VALUES("%s", "%s", %d, "%s")'
\
                  % (cli_id, bank_name, acc_num, acc_type)
            sql2 = 'INSERT INTO Cli_Acc VALUES ("%s", %d)' % (cli_id, acc_num)
            sql_list.append(sql1)
            sql_list.append(sql2)
        elif 'del_cli' in request.form:
            acc_num = int(request.form["acc_num"])
            cli_id = request.form["cli_id"]
            sql_get = 'SELECT COUNT(*) FROM Cli_Acc \
                     WHERE acc_num = %d' % (acc_num)
            try:
                cursor.execute(sql_get)
                sum = cursor.fetchall()[0][0]
                sum = int(sum)
                if sum == 0:
                    return render_template("acc/acc_alt_fail.html")
            except:
                return render_template("acc/acc_alt_fail.html")
            sql1 = 'DELETE FROM Cli_Bank_Acctype \
                   WHERE cli_id = "%s" and acc_num = %d' % (cli_id, acc_num)
            sql2 = 'DELETE FROM Cli_Acc \
                   WHERE cli_id = "%s" and acc_num = %d' % (cli_id, acc_num)
            sql_list.append(sql1)
            sql_list.append(sql2)
        try:
            for sql in sql_list:
                cursor.execute(sql)
                db.commit()
            return redirect(url_for('table'))
        except:
            db.rollback()
```

```
            return render_template("acc/acc_alt_fail.html")
```

查询账户：

可以根据账户号、支行名、客户身份证号查询部分账户，或查询所有账户。

关键代码：

```
if request.method == "POST":
    if 'return' in request.form:
        return redirect(url_for('table'))
    else:
        sql = ''
        if 'by_acc_num' in request.form:
            acc_num = int(request.form["acc_num"])
            sql = 'SELECT * FROM Accounts, Cli_Acc \
                  WHERE Accounts.acc_num = Cli_Acc.acc_num \
                  and Accounts.acc_num = "%d"' % (acc_num)
        elif 'by_bank' in request.form:
            bank_name = request.form["bank_name"]
            sql = 'SELECT * FROM Accounts, Cli_Acc \
                  WHERE Accounts.acc_num = Cli_Acc.acc_num \
                  and Accounts.bank_name = "%s"' % (bank_name)
        elif 'by_cli_id' in request.form:
            cli_id = request.form["cli_id"]
            sql = 'SELECT * FROM Accounts, Cli_Acc \
                  WHERE Accounts.acc_num = Cli_Acc.acc_num \
                  and cli_id = "%s"' % (cli_id)
        elif 'search_all' in request.form:
            sql = 'SELECT * FROM Accounts, Cli_Acc \
                  WHERE Accounts.acc_num = Cli_Acc.acc_num'
        try:
            cursor.execute(sql)
            rows = cursor.fetchall()
#             print(rows)
            return render_template('acc/acc_search.html', rows = rows)
        except MySQLdb.Error as e:
            return render_template('acc/acc_search.html', rows = '')
```

3.4 贷款管理模块

添加贷款：

贷款号由系统自动生成。

关键代码：

```
if request.method == "POST":
    if 'return' in request.form:
        return redirect(url_for('table'))
    elif 'debt_add' in request.form:
        sql_debts = ""
        bank_name = '"' + request.form["bank_name"] + '"'
        amount = float(request.form["amount"])
        cli_id = ["", "", "", "", ""]
        if request.form["cli_id1"] == "":
            cli_id[0] = None
        else:
            cli_id[0] = '"' + request.form["cli_id1"] + '"'
        ……
        if request.form["cli_id5"] == "":
            cli_id[4] = None
        else:
            cli_id[4] = '"' + request.form["cli_id5"] + '"'

        sql_debts = 'INSERT INTO Debts VALUES(null, %s, %f, 0.0)'\
                    % (bank_name, amount)
        try:
            cursor.execute(sql_debts)
            debt_num = db.insert_id()
            debt_num = int(debt_num)
            for i in range(5):
                if cli_id[i] != None:
                    sql_cli_debt = 'INSERT INTO Cli_Debt VALUES(%s, %d)'\
                                % (cli_id[i], debt_num)
                    print(sql_cli_debt)
                    cursor.execute(sql_cli_debt)
            db.commit()
            return redirect(url_for('table'))
        except:
            db.rollback()
```

```
            return render_template("debt/debt_add_fail.html")
```

删除贷款：

需要输入贷款号和与该贷款关联的一位客户的身份证号才能删除贷款。

未发放完的贷款不能删除。

关键代码：

```
if request.method == "POST":
    if 'return' in request.form:
        return redirect(url_for('table'))
    elif 'debt_del' in request.form:
        debt_num = int(request.form["debt_num"])
        cli_id = request.form["cli_id"]
        sql_get = 'SELECT amount, paid_amount FROM Debts \
                  WHERE debt_num = %d' % (debt_num)
        print(sql_get)
        try:
            cursor.execute(sql_get)
            amount, paid_amount = cursor.fetchall()[0]
            amount = float(amount)
            paid_amount = float(paid_amount)
            if amount != paid_amount:
                return render_template("debt/debt_del_fail.html")
        except:
            return render_template("debt/debt_del_fail.html")
        sql1 = 'DELETE FROM Debts_pay WHERE debt_num = %d' % (debt_num)
        sql2 = 'DELETE FROM Cli_Debt WHERE debt_num = %d' % (debt_num)
        sql3 = 'DELETE FROM Debts WHERE debt_num = %d' % (debt_num)
        try:
            cursor.execute(sql1)
            cursor.execute(sql2)
            cursor.execute(sql3)
            db.commit()
            return redirect(url_for('table'))
        except:
            db.rollback()
            return render_template("debt/debt_del_fail.html")
```

查询贷款：

根据贷款号、客户身份证号或支行名查询贷款，或查询全部贷款。

关键代码：

```
    if request.method == "POST":
        if 'return' in request.form:
            return redirect(url_for('table'))
        else:
            sql = ''
            if 'by_debt_num' in request.form:
                debt_num = int(request.form["debt_num"])
                sql = 'SELECT Debts.debt_num, bank_name, amount, paid_amount, cli_id
\
                      FROM Debts, Cli_Debt \
                      WHERE Debts.debt_num = Cli_Debt.debt_num \
                      and Debts.debt_num = %d' % (debt_num)
            elif 'by_cli_id' in request.form:
                cli_id = request.form["cli_id"]
                sql = 'SELECT Debts.debt_num, bank_name, amount, paid_amount, cli_id
\
                      FROM Debts, Cli_Debt \
                      WHERE Debts.debt_num = Cli_Debt.debt_num \
                      and cli_id = "%s"' % (cli_id)
            elif 'by_bank_name' in request.form:
                bank_name = request.form["bank_name"]
                sql = 'SELECT Debts.debt_num, bank_name, amount, paid_amount, cli_id
\
                      FROM Debts, Cli_Debt \
                      WHERE Debts.debt_num = Cli_Debt.debt_num \
                      and bank_name = "%s"' % (bank_name)
            elif 'search_all' in request.form:
                sql = 'SELECT Debts.debt_num, bank_name, amount, paid_amount, cli_id
\
                      FROM Debts, Cli_Debt \
                      WHERE Debts.debt_num = Cli_Debt.debt_num'
            try:
                cursor.execute(sql)
                rows = cursor.fetchall()
                return render_template('debt/debt_search.html', rows = rows)
            except MySQLdb.Error as e:
                return render_template('debt/debt_search.html', rows = '')
```

发放贷款：

关键代码：

```
if request.method == "POST":
    if 'return' in request.form:
        return redirect(url_for('table'))
    elif 'debt_pay' in request.form:
        debt_num = int(request.form["debt_num"])
        pay_date = '"' + time.strftime("%Y-%m-%d", time.localtime()) + '"'
        pay_sum = float(request.form["pay_sum"])
        sql_get = 'SELECT amount, paid_amount FROM Debts WHERE debt_num = %d' \
                  % (debt_num)
        try:
            cursor.execute(sql_get)
            amount, paid_amount = cursor.fetchall()[0]
            amount = float(amount)
            paid_amount = float(paid_amount)
        except:
            return render_template("debt/debt_pay_fail.html")
        if pay_sum <= 0 or pay_sum+paid_amount > amount:
            return render_template("debt/debt_pay_fail.html")
        sql1 = 'INSERT INTO Debts_pay VALUES(%d, %s, %f)' \
               % (debt_num, pay_date, pay_sum)
        sql2 = 'UPDATE Debts SET paid_amount = %f WHERE debt_num = %d' \
               % (pay_sum+paid_amount, debt_num)
        try:
            cursor.execute(sql1)
            cursor.execute(sql2)
            db.commit()
            return redirect(url_for('table'))
        except:
            db.rollback()
            return render_template("debt/debt_pay_fail.html")
```

3.5 业务统计模块

使用 echarts 生成图表。

按类型：

根据贷款数、储蓄账户数、支票账户数生成饼图。

关键代码：

main.py

```
    else:
        sql1 = 'SELECT COUNT(*) FROM Save_accounts'
        sql2 = 'SELECT COUNT(*) FROM Check_accounts'
        sql3 = 'SELECT COUNT(*) FROM Debts'
        try:
            cursor.execute(sql1)
            cnt = cursor.fetchall()[0][0]
            cnt_save = int(cnt)
            cursor.execute(sql2)
            cnt = cursor.fetchall()[0][0]
            cnt_check = int(cnt)
            cursor.execute(sql3)
            cnt = cursor.fetchall()[0][0]
            cnt_debt = int(cnt)
            return render_template('/by_types.html', cnt_save=cnt_save,
cnt_check=cnt_check, cnt_debt=cnt_debt)
        except:
            return render_template('/by_types.html', cnt_save=0, cnt_check=0,
cnt_debt=0)
```

by_types.html

```
    <script type="text/javascript">
        var myChart = echarts.init(document.getElementById('main'));
        var option = {
            title : {
                text: '业务统计-按类型',
                x:'center'
            },
            tooltip : {
                trigger: 'item',
                formatter: "{a} <br/>{b} : {c} ({d}%)"
            },
            legend: {
                orient: 'vertical',
                left: 'left',
                data: ['存款账户','支票账户','贷款']
            },
            series : [
                {
```

```
                name: '数量',
                type: 'pie',
                radius : '55%',
                center: ['50%', '50%'],
                data:[
                    {value:{{ cnt_save }}, name:'存款账户'},
                    {value:{{ cnt_check }}, name:'支票账户'},
                    {value:{{ cnt_debt }}, name:'贷款'}
                ],
                itemStyle: {
                    emphasis: {
                        shadowBlur: 10,
                        shadowOffsetX: 0,
                        shadowColor: 'rgba(0, 0, 0, 0.5)'
                    }
                }
            }
        ],
        color:
['rgb(131,175,155)','rgb(252,157,154)','rgb(249,205,173)','rgb(200,200,169)',]
    };
    myChart.setOption(option);
</script>
```

按时间：

根据日期和开户、发放贷款情况生成折线图。

关键代码：

by_time.py

```
if request.method == "POST":
    return redirect(url_for('table'))
else:
    sql_acc = 'SELECT open_date, COUNT(*) FROM Accounts \
              GROUP BY open_date'
    sql_debt = 'SELECT pay_date, COUNT(*) FROM Debts_pay \
              GROUP BY pay_date'
    try:
        cursor.execute(sql_acc)
        acc_rows = cursor.fetchall()
        print(acc_rows)
```

```
            cursor.execute(sql_debt)
            debt_rows = cursor.fetchall()
            print(debt_rows)
            return render_template('/by_time.html', acc_rows=acc_rows,
debt_rows=debt_rows)
        except:
            return render_template('/by_time.html', acc_rows='', debt_rows='')
```

by_time.html

```
    <script type="text/javascript">
        var myChart = echarts.init(document.getElementById('acc'));
        var option = {
            dataZoom: [{
                        type: 'slider',
                        show: true,
                        xAxisIndex: [0],
                        left: '9%',
                        bottom: -5,
                        start: 10,
                        end: 20
                    }],
            xAxis: {
                name: "日期",
                type: 'category',
                data:[
                     {% for row in acc_rows %}
                         "{{ row[0]}}",
                     {% endfor %}
                   ]},
            yAxis: {
                name:"数量",
                type: 'value',
                axisLabel : {formatter: '{value}'}
            },
            series: [{
                type: 'line',
                data:[
                    {% for row in acc_rows %}
                        "{{ row[1] }}",
                    {% endfor %}
                ]
            }],
```

```
    };
    myChart.setOption(option);
</script>
```

4 实现与测试

4.1 实现结果

主页：

客户管理：


添加客户
身份证号
姓名
联系电话
家庭住址
联系人姓名
联系人手机号
联系人Email
联系人关系
联系银行员工
添加客户
返回


删除客户
身份证号
删除客户
返回


修改客户信息
修改个人信息
客户身份证号
联系电话
家庭住址
修改个人信息
修改联系人信息
客户身份证号
联系人姓名
联系人手机号
联系人Email
联系人关系
修改联系人信息


查询客户
身份证号 姓名 联系电话 家庭住址 联系人姓名 联系人手机号 联系人Email 联系人关系 联系银行员工
身份证号
查询客户
姓名
查询客户
查询全部
返回


修改联系员工
客户身份证号
联系银行员工
修改联系员工
返回

账户管理：


开户
支行名
客户1身份证号
客户2身份证号
客户3身份证号
客户4身份证号
客户5身份证号
账户类型
●储蓄账户 ○支票账户
利率（储蓄账户）
货币类型（储蓄账户）
●1 ○2 ○3 ○4
透支额（支票账户）
开户
返回


销户
账户号
客户身份证号
销户
返回


修改账户
存款
账户号
金额
存款
取款
账户号
客户身份证号
金额
取款
修改支行
账户号
支行名
修改支行


查询账户
账户号
支行名
余额
开户日期
访问日期
类型
客户身份证号
账户号
查询账户
支行名
查询账户
客户身份证号
查询账户
查询全部
返回


添加账户
账户号
客户身份证号
添加客户
删除账户
账户号
客户身份证号
删除客户
返回

贷款管理：

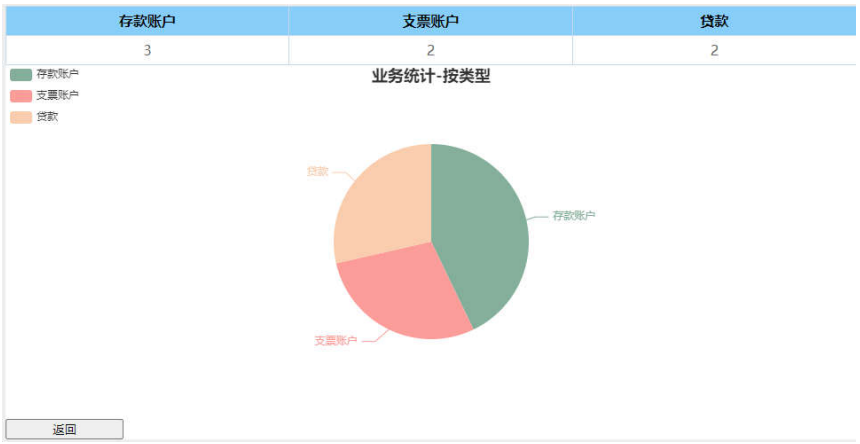

添加贷款
支行名
金额
客户1身份证号
客户2身份证号
客户3身份证号
客户4身份证号
客户5身份证号
添加贷款
返回


删除贷款
贷款号
客户身份证号
删除贷款
返回


查询贷款
贷款号
支行名
金额
已支付金额
客户身份证号
贷款号
查询贷款
客户身份证号
查询贷款
支行名
查询贷款
查询全部
返回

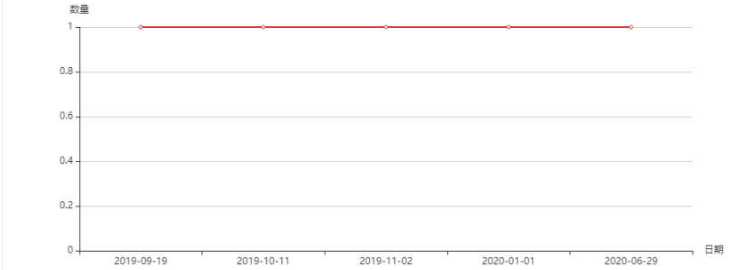

发放贷款
贷款号
金额
发放贷款
返回

业务统计：


存款账户
支票账户
贷款
3
2
2
业务统计-按类型
存款账户
支票账户
贷款
返回

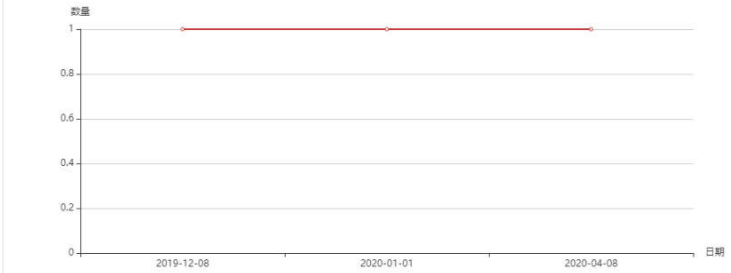
开户统计

日期	数量
2019-09-19	1
2019-10-11	1
2019-11-02	1
2020-01-01	1
2020-06-29	1



贷款统计

日期	数量
2019-12-08	1
2020-01-01	1
2020-04-08	1



返回

4.2 测试结果

客户管理模块：

查询客户

身份证号	姓名	联系电话	家庭住址	联系人姓名	联系人手机号	联系人Email	联系人关系	联系银行员工
001	name1	00000000001	addr1	con1	00000000011	001@e.com	relat1	001
002	name'2	00000000002	addr2	con2	00000000022	002@e.com	relat2	002
003	name3	00000000003	addr1	con3	00000000033	003@e.com	relat1	001
004	name4	00000000004	addr2	con4	00000000044	004@e.com	relat2	None

身份证号

查询客户

姓名

查询客户

查询全部

返回

账户管理模块：

贷款管理模块：

业务统计模块：

见 4.1 实现结果

5 总结与讨论

感谢助教提供的模板。