

# “银行业务管理系统”

## 系统设计与实现报告

姓名：彭博

学号：PB19071463

计算机科学与技术学院

中国科学技术大学

2022 年 7 月

## 目 录

1 概述 .....	1
1.1 系统目标 .....	1
1.2 需求说明 .....	1
1.3 本报告的主要贡献 .....	2
2 总体设计 .....	2
2.1 系统模块结构 .....	2
2.2 系统工作流程 .....	4
2.3 数据库设计 .....	4
3 详细设计【可选】 .....	5
3.1 **** 模块 .....	错误!未定义书签。
3.2 **** 模块 .....	错误!未定义书签。
3.3 **** 模块 .....	错误!未定义书签。
4 实现与测试 .....	5
4.1 实现结果 .....	5
4.2 测试结果 .....	6
4.3 实现中的难点问题及解决【可选】 .....	20
5 总结与讨论 .....	21

# 1 概述

## 1.1 系统目标

开发一个银行业务管理系统

## 1.2 需求说明

### 数据需求：

银行有多个支行。各个支行位于某个城市，每个支行有唯一的名字。银行要监控每个支行的资产。银行的客户通过其身份证号来标识。银行存储每个客户的姓名、联系电话以及家庭住址。为了安全起见，银行还要求客户提供一位联系人的信息，包括联系人姓名、手机号、Email 以及与客户的关系。客户可以有帐户，并且可以贷款。客户可能和某个银行员工发生联系，该员工是此客户的贷款负责人或银行帐户负责人。银行员工也通过身份证号来标识。员工分为部门经理和普通员工，每个部门经理都负责领导其所在部门的员工，并且每个员工只允许在一个部门内工作。每个支行的管理机构存储每个员工的姓名、电话号码、家庭地址、所在的部门号、部门名称、部门类型及部门经理的身份证号。银行还需知道每个员工开始工作的日期，由此日期可以推知员工的雇佣期。银行提供两类帐户——储蓄帐户和支票帐户。帐户可以由多个客户所共有，一个客户也可开设多个账户，但在一个支行内最多只能开设一个储蓄账户和一个支票账户。每个帐户被赋以唯一的帐户号。银行记录每个帐户的余额、开户日期、开户的支行名以及每个帐户所有者访问该帐户的最近日期。另外，每个储蓄帐户有利率和货币类型，且每个支票帐户有透支额。每笔贷款由某个分支机构发放，能被一个或多个客户所共有。每笔贷款用唯一的贷款号标识。银行需要知道每笔贷款所贷金额以及逐次支付的情况（银行将贷款分几次付给客户）。虽然贷款号不能唯一标识银行所有为贷款所付的款项，但可以唯一标识为某贷款所付的款项。对每次的付款需要记录日期和金额

### 管理需求：

**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（储蓄、 贷款）和时间（月、 季、 年）统计各个支行的业务总金额和用户数， 统计的结果以表格形式展示。

### 1.3 本报告的主要贡献

介绍了基于 MySQL 数据库、 Flask 的 B/S 架构的数据库系统的基本开发流程，包括实现用到的语言、 开发工具、 系统模块及作用、 前后端交互等等

## 2 总体设计

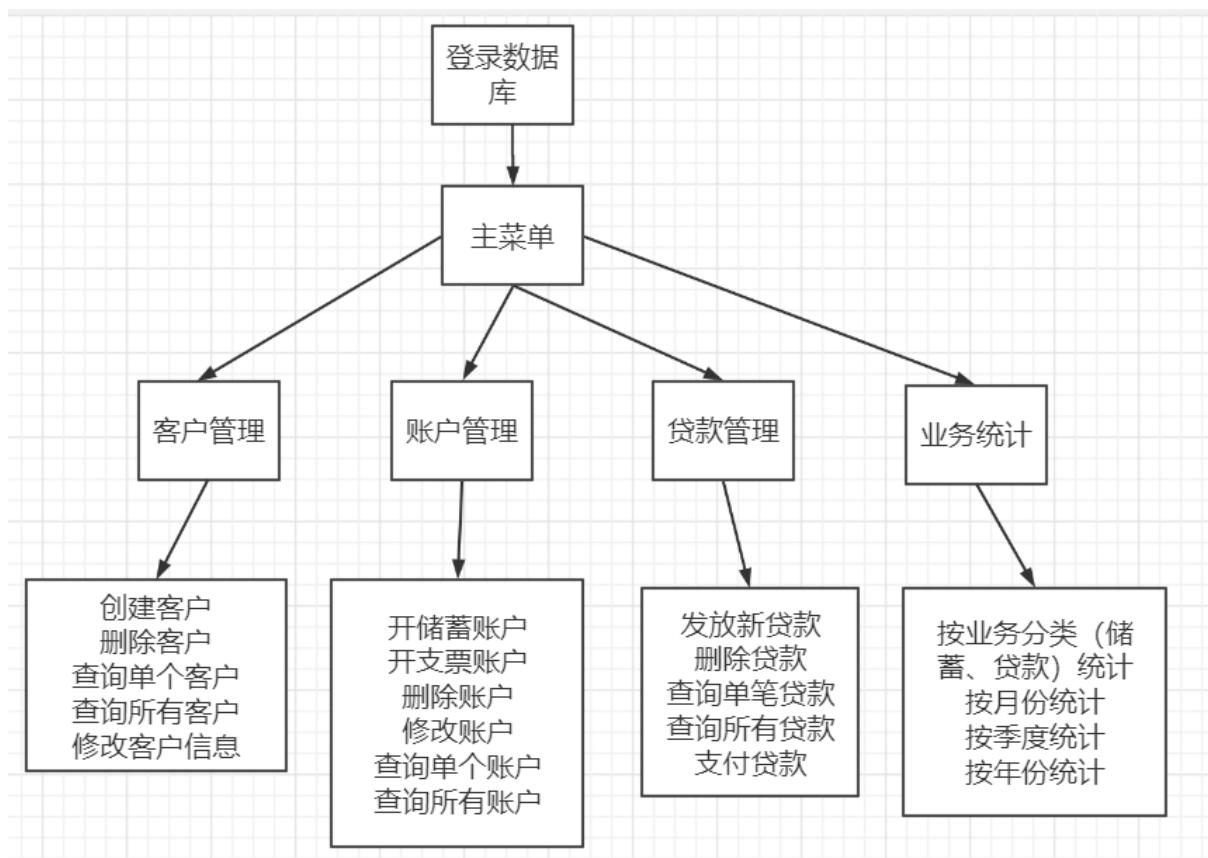
### 2.1 系统模块结构

```
bankSys\
|---sql\
    |---//用以初始化数据库的 sql 语句
    |---demo.sql
|---static\
    |---//静态文件， 即助教给出的 demo 中的文件
    |---404.png
    |---style.css
|---templates
    |---account ---//账户管理
        |---menu.html---//账户管理菜单
        |---checking_account.html---//开支票账户
        |---saving_account.html---//开储蓄账户
        |---delete.html---//删除账户
        |---update.html---//修改账户
        |---search_input.html---//输入要搜索的单个账户的 ID
        |---search_single.html---//搜索单个账户
        |---search.html---//搜索所有账户
    |---customer--- //客户管理
        |---menu.html---//客户管理菜单
```

---

```
|---insert.html---//增加客户
|---delete.html---//删除客户
|---update.html---//修改客户信息
|---search_input.html---//输入要搜索的单个客户的 ID
|---search_single.html---//搜索单个客户
|---search.html---//搜索所有客户
|---loan ---//贷款管理
    |---menu.html---//贷款管理菜单
    |---create.html---//发放贷款
    |---delete.html---//删除贷款
    |---pay_loan.html---//支付贷款
    |---search_input.html---//输入要搜索的单笔贷款的 ID
    |---search_single.html---//搜索单笔贷款
    |---search.html---//搜索所有贷款
|---statistics---//业务统计
    |---menu.html---//业务统计菜单
    |---classify.html---//按照业务分类（储蓄、贷款）统计
    |---month.html---//按月份统计
    |---quarter.html---//按季度统计
    |---year.html---//按年份统计
|---404.html---//找不到相关页面
|---login.html---//登录界面
|---login_fail.html---//登录失败界面
|---menu.html---//主菜单
|---app.py---//运行文件
|---db.py---//与数据库进行交互的接口
```

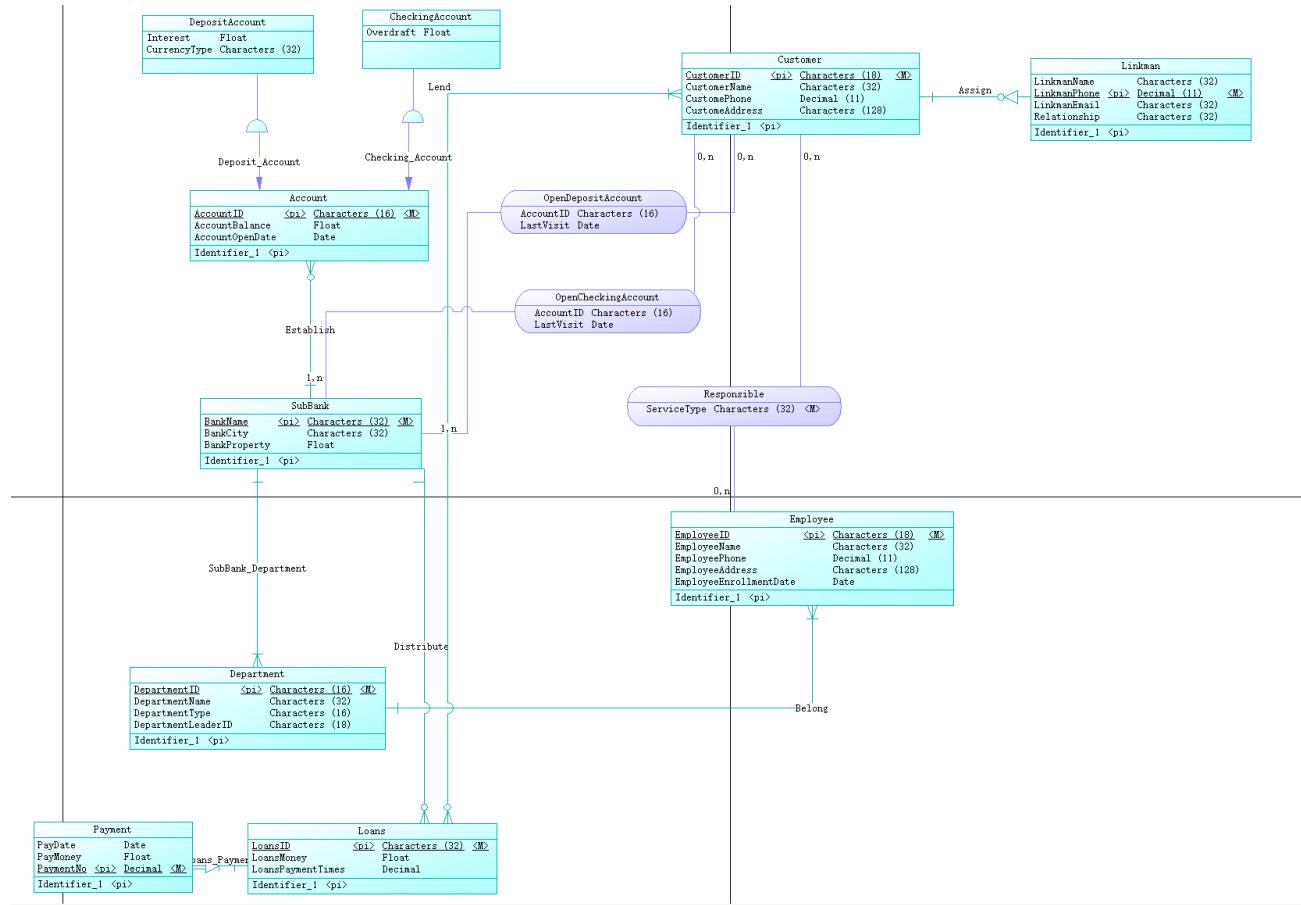
## 2.2 系统工作流程



## 2.3 数据库设计

使用 PowerDesign16 工具生成

ER 图如下



详细的物理数据库设计参见附件中的 demo.sql (参考了往届助教给出的模板 sql)

### 3 详细设计

### 4 实现与测试

#### 4.1 实现结果

这里主要展示各个菜单的界面，具体功能界面及结果在 4.2 中给出  
具体实现参见附件中的代码

主菜单



### 客户管理菜单



### 账户管理菜单



贷款管理菜单



## 业务统计



## 4.2 测试结果

### 客户管理

#### 创建客户

[添加客户](#)

客户身份账号

111

客户姓名

pqz

客户联系电话

150

客户家庭住址

ustc

联系人电话

135

联系人姓名

plo

联系人Email

pqz@ustc.com

联系人与客户关系

friend

[插入](#)

查询所有客户，可以看到多出如下表项

Customer表中的所有记录

客户身份证	客户姓名	客户联系电话	客户家庭住址	联系人电话	联系人姓名	联系人Email	联系人与客户关系
111	pqz	150	ustc	135	plo	pqz@ustc.com	friend
1234	1234	1234	1234	1234	1234	1234	1234
1314	LucyHe	123	earch	135	Pqz	789@1.com	friend
369	plm	11	11	11	11	11	11
963	123	123	123	123	123	123	123
pqz	pqz	pqz	pqz	pqz	p	p	p

[回到上一级菜单](#)

插入时会检查客户名称中是否含有单引号，如果含有则无法插入

#### 删除客户

删除无关联账户或贷款记录的客户，ID 为 1314

## 删除客户

**客户身份证号**



可以看到删除成功

## Customer表中的所有记录

客户身份证	客户姓名	客户联系电话	客户家庭住址	联系人电话	联系人姓名	联系人Email	联系人与客户关系
111	pqz	150	ustc	135	plo	pqz@ustc.com	friend
1234	1234	1234	1234	1234	1234	1234	1234
369	plm	11	11	11	11	11	11
963	123	123	123	123	123	123	123
pqz	pqz	pqz	pqz	pqz	p	p	p

## 查询单个客户

查询我们刚才插入的 ID 为 111 的客户

### 查询单个客户

**客户身份证号**



结果如下

### 查询结果

客户身份证	客户姓名	客户联系电话	客户家庭住址	联系人电话	联系人姓名	联系人Email	联系人与客户关系
111	pqz	150	ustc	135	plo	pqz@ustc.com	friend

## 查询所有客户

## Customer表中的所有记录

客户身份证号	客户姓名	客户联系电话	客户家庭住址	联系人电话	联系人姓名	联系人Email	联系人与客户关系
111	pqz	150	ustc	135	plo	pqz@ustc.com	friend
1234	1234	1234	1234	1234	1234	1234	1234
369	plm	11	11	11	11	11	11
963	123	123	123	123	123	123	123
pqz	pqz	pqz	pqz	pqz	p	p	p

[回到上一级菜单](#)

## 修改客户信息

我们将刚才插入的 ID 为 111 的客户的基本信息作如下修改

## 输入客户身份证号

客户身份证号

客户姓名

客户联系电话

客户家庭住址

联系人电话

联系人姓名

联系人Email

联系人与客户关系

[修改](#)

修改后结果如下

## Customer表中的所有记录

客户身份证号	客户姓名	客户联系电话	客户家庭住址	联系人电话	联系人姓名	联系人Email	联系人与客户关系
111	pqzzz	150150	USTC	135135	ppo	ppo@123.com	teacher
1234	1234	1234	1234	1234	1234	1234	1234
369	plm	11	11	11	11	11	11
963	123	123	123	123	123	123	123
pqz	pqz	pqz	pqz	pqz	p	p	p

[回到上一级菜单](#)

## 账户管理

## 开储蓄账户

为客户 ID 为 111 的客户开一个储蓄账户

## 开储蓄账户

## 客户身份证号

## 账户号

## 余额

## 开户时间

## 利率

## 货币类型

[开户](#)

结果如下，其最近访问时间为开户时间

## Account表中所有记录

账户号	客户身份证号	账户类型	最近访问时间
111	111	saving	2018-01-01 00:00:00
369	369	saving	2021-07-06 11:54:41
999	1234	saving	2021-07-06 11:47:34

[回到上一级菜单](#)

## 开支票账户

为客户 ID 为 111 的客户开一个支票账户

**开支票账户**

---

<b>客户身份证号</b>	<input type="text" value="111"/>
<b>账户号</b>	<input type="text" value="567"/>
<b>余额</b>	<input type="text" value="567"/>
<b>开户日期</b>	<input type="text" value="2020-1-1"/>
<b>透支额</b>	<input type="text" value="3000"/>
<input type="button" value="开户"/>	

结果如下，其最近访问时间会被初始化为开户时间

Account表中所有记录

账户号	客户身份证号	账户类型	最近访问时间
111	111	saving	2018-01-01 00:00:00
369	369	saving	2021-07-06 11:54:41
567	111	checking	2020-01-01 00:00:00
999	1234	saving	2021-07-06 11:47:34

### 删除账户

删除刚才为 ID111 的客户开设的支票账户

**删除账户**

---

<b>账户号</b>	<input type="text" value="567"/>
<input type="button" value="删除"/>	

删除后结果如下

### Account表中所有记录

账户号	客户身份证号	账户类型	最近访问时间
111	111	saving	2018-01-01 00:00:00
369	369	saving	2021-07-06 11:54:41
999	1234	saving	2021-07-06 11:47:34

[回到上一级菜单](#)

### 修改账户

在实现中支持修改余额、利率、货币类型和透支额信息。除了账户号外其他信息的修改的实现其实都大同小异，基本思路都是一样的。

将账户号为 111 的账户作如下修改

请输入合法的账户号

账户号	<input type="text" value="111"/>
余额	<input type="text" value="999"/>
利率	<input type="text" value="1.99"/>
货币类型	<input type="text" value="0"/>
透支额	<input type="text" value="0"/>
<input type="button" value="修改"/>	

由于写的系统只支持查看账户号、客户身份证号、账户类型、最近访问时间信息，故查看 MySQL 中的表格，结果如下

accounts 表

	accountID	money	settime	accounttype
▶	111	999	2018-01-01 00:00:00	saving
	369	10000	2018-01-01 00:00:00	saving
*	999	6	2020-07-01 00:00:00	saving
	NULL	NULL	NULL	NULL

saveacc 表

	accountID	interestrate	savetype
1	111	1.99	0
	369	8	1
	999	5.5	0
*	NULL	NULL	NULL

## 查询单个账户

查询 ID 为 111 的账户信息

### 查询单个账户

账户号

111

搜索

结果如下

### 查询结果

账户号	客户身份证号	账户类型	最近访问时间
111	111	saving	2021-07-06 16:39:05

[回到上一级菜单](#)

可以看到最近访问时间对应作了修改

## 查询所有账户

### Account表中所有记录

账户号	客户身份证号	账户类型	最近访问时间
111	111	saving	2021-07-06 16:39:05
369	369	saving	2021-07-06 11:54:41
999	1234	saving	2021-07-06 11:47:34

[回到上一级菜单](#)

## 贷款管理

贷款的状态位: 0-未开始发放, 1-发放中, 2-已全部发放

### 发放新贷款

由于一笔贷款至少被一个客户拥有，所以在发放时需要指定一个客户

### 发放新贷款

客户身份证号

贷款号

贷款金额

结果如下

### Loan中的所有贷款

贷款号	客户身份证号	贷款金额	状态
1266	1234	88.0	1
321	111	5000.0	0
567	963	111.0	0

### 删除贷款

删除贷款号为 567 的贷款

### 删除贷款

贷款号

结果如下

### Loan中的所有贷款

贷款号	客户身份证号	贷款金额	状态
1266	1234	88.0	1
321	111	5000.0	0

## 查询单笔贷款

查询贷款号为 321 的贷款

**查询单笔贷款**

**贷款号**

**搜索**

结果如下

**查询结果**

贷款号	客户身份证号	贷款金额	状态
321	111	5000.0	0

**回到上一级菜单**

## 查询所有贷款

### Loan中的所有贷款

贷款号	客户身份证号	贷款金额	状态
1266	1234	88.0	1
321	111	5000.0	0

**回到上一级菜单**

## 支付贷款

首先 ID 为 111 的客户为 ID 为 321 的贷款支付 900 元

### 贷款支付

贷款号

客户身份证号

支付金额

支付时间

可以看到对应的状态变为 1，即发放中

### Loan中的所有贷款

贷款号	客户身份证号	贷款金额	状态
1266	1234	88.0	1
321	111	5000.0	1

再让 ID 为 1234 的客户为 ID 为 321 的贷款支付 4100 元

### 贷款支付

贷款号

客户身份证号

支付金额

支付时间

可以看到对应贷款的状态变为 2，即发放完成 ( $900+4100=5000$ )

### Loan中的所有贷款

贷款号	客户身份证号	贷款金额	状态
1266	1234	88.0	1
321	111	5000.0	2

[回到上一级菜单](#)

## 业务统计

### 按业务分类统计（储蓄、贷款）

#### 按业务分类（储蓄、贷款）统计

银行	储蓄总金额	储蓄客户数
888	11005.0	3

银行	贷款总金额	贷款客户数
888	5088.0	2

[回到上一级菜单](#)

增加一些数据，然后进行如下统计

### 按月份统计

#### 按月份统计

月份	业务总金额	业务用户数
202107	11005.0	3
201806	987.0	1
201501	1255.0	1

[回到上一级菜单](#)

### 按季度统计（xxxx\_y 表示 xxxx 年的第 y 季度）

### 按季度统计

季度	业务总金额	业务用户数
2021_3	11005.0	3
2018_2	987.0	1
2015_1	1255.0	1

[回到上一级菜单](#)

### 按年份统计

### 按年份统计

年份	业务总金额	业务用户数
2021	11005.0	3
2018	987.0	1
2015	1255.0	1

[回到上一级菜单](#)

## 4.3 实现中的难点问题及解决

如果账户客户存在关联账户或贷款记录则无法删除，这是由表之间的约束保证的。具体来说

1. cusforacc 表外键依赖于 customer 表中的主键 cusID，如果该客户有关联账户，则 cusforacc 表中就会有对应的表项。如果删除 customer 表中的对应项会违反参照完整性。所以在有关联客户时无法删除客户信息。
2. loan 表同理外键依赖于 customer 表中的主键 cusID，如果删除 loan 表中的对应项也会违反参照完整性。所以在贷款记录时无法删除客户信息。

业务统计时需要用到使用特殊的 sql 函数，根据时间粒度进行分组。比如按月统计：

```
SELECT DATE_FORMAT(visit, '%Y%m') months, SUM(money), COUNT(accounts.accountID)
FROM cusforacc, accounts
WHERE cusforacc.accountID = accounts.accountID
GROUP BY months;
```

删除处于发放中的、贷款号为 1266 的贷款是不被允许的，这通过 sql 定义的如下触发器实现。贷款状态的变化也是通过触发器来实现的。具体参见 demo.sql

## 5 总结与讨论

从本次实验中，我对基于 Python Flask、MySQL 的 B/S 架构的数据库开发有了深入的了解。

学到的经验和教训有很多。

首先是要对自己即将使用的工具有个入门级别的了解，心中要有一个大体的框架，然后再细化到每一个模块，查找官方文档来编码。模块化是一个很重要的思想，我的项目中其实主要是两部分，第一部分是用来和 MySQL 交互的，第二部分是用来和前端网页交互并调用第一部分中的 API。具体每个部分又根据功能的需求划分为更多的子模块。

同时在 debug 时也要遵循模块化的思想，每实现一个小的模块、功能就 debug，防止代码过多时难以定位 bug 的位置。

由于时间有限，前端主要参考助教给出的 demo，进行了简单的扩展，主要着重于功能，界面并不是很美观，后续可以考虑结合一些前端框架如 bootstrap 等，形成一个更美观的 UI 界面。