哈工大计算机学院系统安全

实验三

完整性访问控制系统设计与实现

学号：1160300906

姓名：姜梦奇

指导老师：张玥

时间：2018 年 12 月 26 日

一．实验内容

设计完整性访问控制系统，实现系统，并满足某商业公司的完整性访问控制需求。

（1）配合第 7 章，为商业公司设计系统，提出该应用系统的安全策略。

（2）配合第 9 章 为商业公司设计系统，应用系统满足完整性需求。

（3）具体指明是哪类应用系统，应用背景范围不限，可以是银行、股票等，符合商业系统完整性需求即可。

（4）8 学时，每人独立完成。

二．实验环境

Windows 10 ; eclipse ; MySql 8.0

三．实验设计

3.1 数据库的设计

1.建立一个数据库

```
mysql> create database test;
Query OK, 1 row affected (0.17 sec)
```

2. 指定 test 为当前要操作的数据库

```
mysql> use test
Database changed
```

3. 创建顾客信息表 t_user;

Accountnumber 表示用户的账户；Username 表示用户的姓名；password 表示用户的密码；

Balance 表示用户账户当前余额；

```
mysql> create table t_user(accountnumber int AUTO_INCREMENT,
    -> username varchar(20),password varchar(20),balance float,
    -> PRIMARY KEY(accountnumber));
Query OK, 0 rows affected (1.00 sec)

mysql> desc t_user;
+---------------+-------------+------+-----+---------+----------------+
| Field         | Type        | Null | Key | Default | Extra          |
+---------------+-------------+------+-----+---------+----------------+
| accountnumber | int(11)     | NO   | PRI | NULL    | auto_increment |
| username      | varchar(20) | YES  |     | NULL    |                |
| password      | varchar(20) | YES  |     | NULL    |                |
| balance       | float       | YES  |     | NULL    |                |
+---------------+-------------+------+-----+---------+----------------+
4 rows in set (0.39 sec)
```

4. 创建历史信息记录表 t_history;

历史信息记录表是就是记录日志信息的表；

Hid 表示记录的 id，即表示第几条日志；accountnumber 表示进行了操作的用户的账户的 id；

Money 表示操作中变动的钱；content 表示操作的类型，有开户、取款、存款、转账等。

```
mysql> desc t_history;
+---------------+-------------+------+-----+---------+----------------+
| Field         | Type        | Null | Key | Default | Extra          |
+---------------+-------------+------+-----+---------+----------------+
| hid           | int(11)     | NO   | PRI | NULL    | auto_increment |
| accountnumber | int(11)     | YES  |     | NULL    |                |
| money         | float       | YES  |     | NULL    |                |
| content       | varchar(20) | YES  |     | NULL    |                |
| time          | varchar(30) | YES  |     | NULL    |                |
+---------------+-------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
```

5. 创建管理员信息表 t_admin;

管理员信息表里的内容是固定的，不同的管理员可以执行不同的操作，每个管理员承担相应的职责；

Admin1 的职责是查看日志，只有 admin1 身份可以查看日志，即整个系统中所有人的存取款、开户的信息；

Admin2 的职责是可以查看整个用户表，获取用户的余额信息，可以查看所有人的余额，但不能查看密码，余额和密码信息具有敏感性，我们应该保护一部分的信息；

Admin3 的职责是可以查看所有用户的密码，但不能查看余额，理由同上；

```
mysql> select * from t_admin;
+--------+--------+
| name   | passwd |
+--------+--------+
| admin1 | 123456 |
| admin2 | 223456 |
| admin3 | 323456 |
+--------+--------+
```

6. 创建员工信息表 t_worker;

```
mysql> select * from t_worker;
+------------+----------+
| WorkerName | password |
+------------+----------+
| aaa        | aaaaaa   |
| bbb        | bbbbbb   |
+------------+----------+
2 rows in set (0.00 sec)
```

3.2 安全策略文档

安全策略将系统状态划分为:

- 被授权状态（安全）

前端：

1. 普通工作人员可以与用户交互，进行存取款，转账等操作；
2. Admin1 可以对日志进行审查；
3. Admin2 可以对余额进行查看；
4. Admin3 可以对密码进行查看；

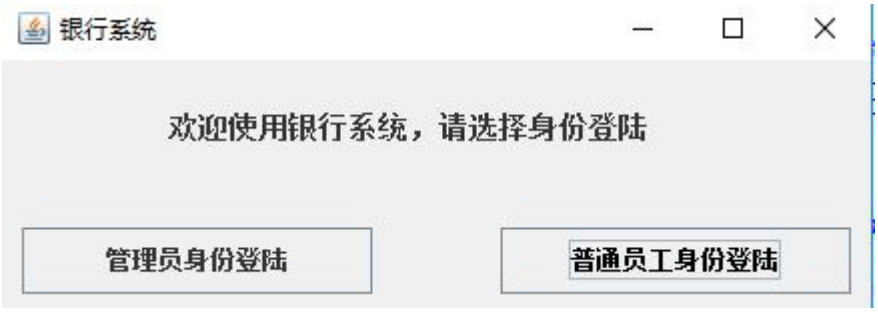
后端：

5. 后端的 root 可以管理整个数据库；
6. backup 可以对数据库进行备份；
7. monitor 监控数据库服务器，维护安全；
8. security 创建、管理用户，设置权限，查看日志

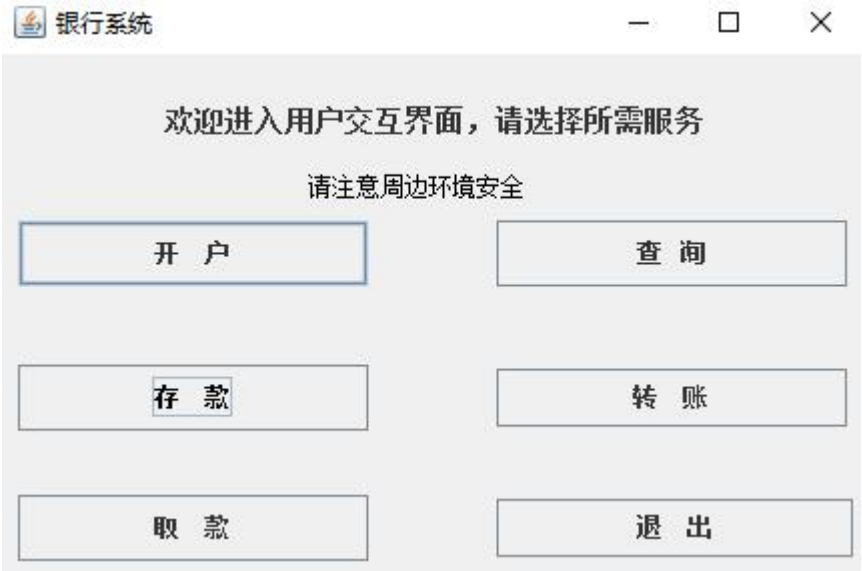
- 未授权状态（不安全），系统进入这些状态，违背了安全性

不满足上面 5 条的状态都是不安全的；比如一个管理员同时看到用户密码与余额；比如不被允许的非法存取款，转账操作都是不安全的；

3.3 提供交互界面，能够完成录入、查询等功能。



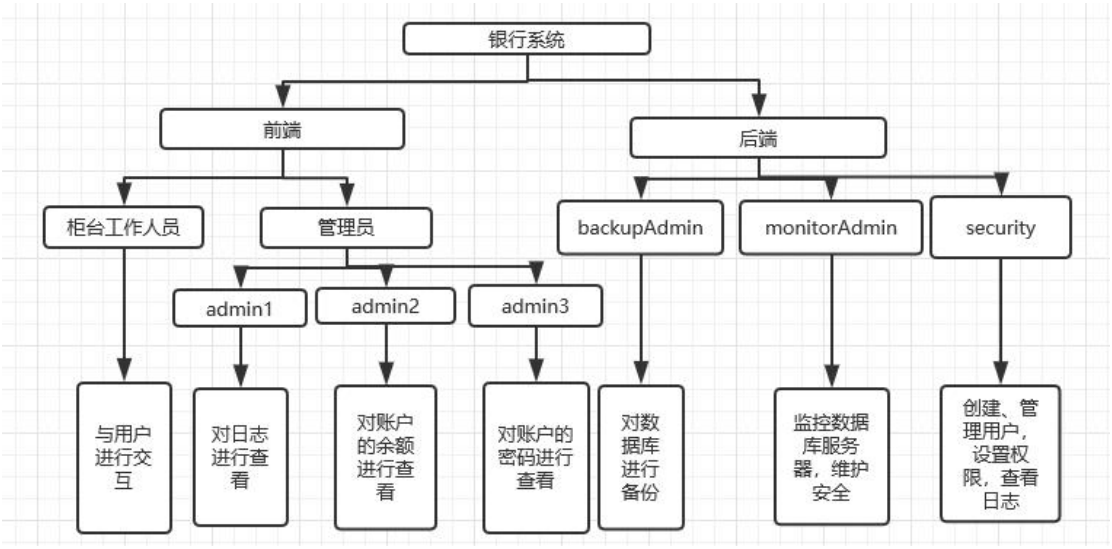
选择普通员工身份登陆，可以进行与用户的交互：

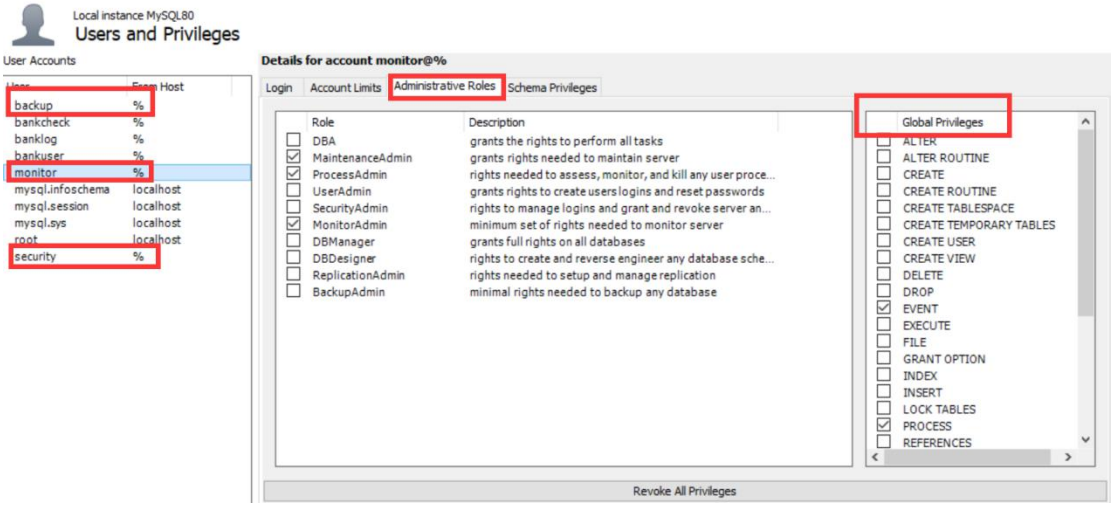


以上各种服务的界面截图：

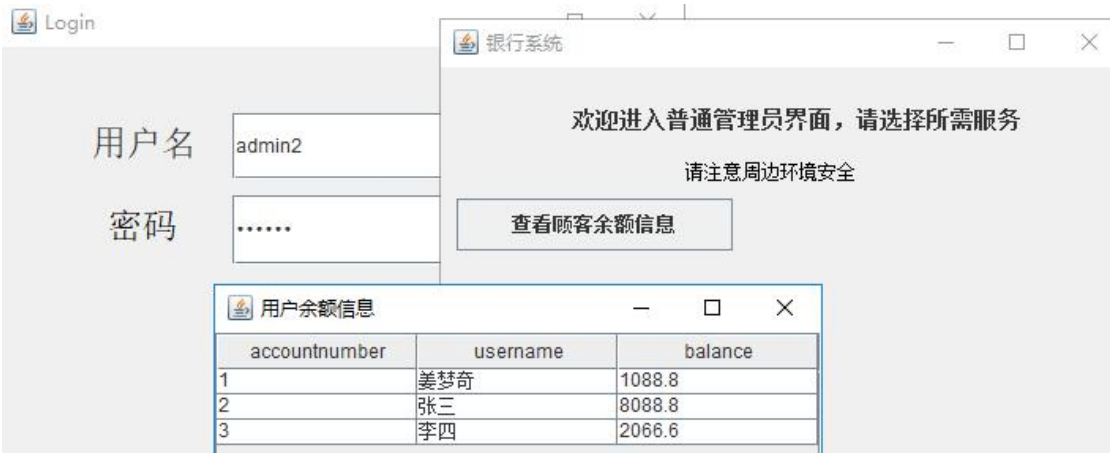
3.4 满足责任分离原则。

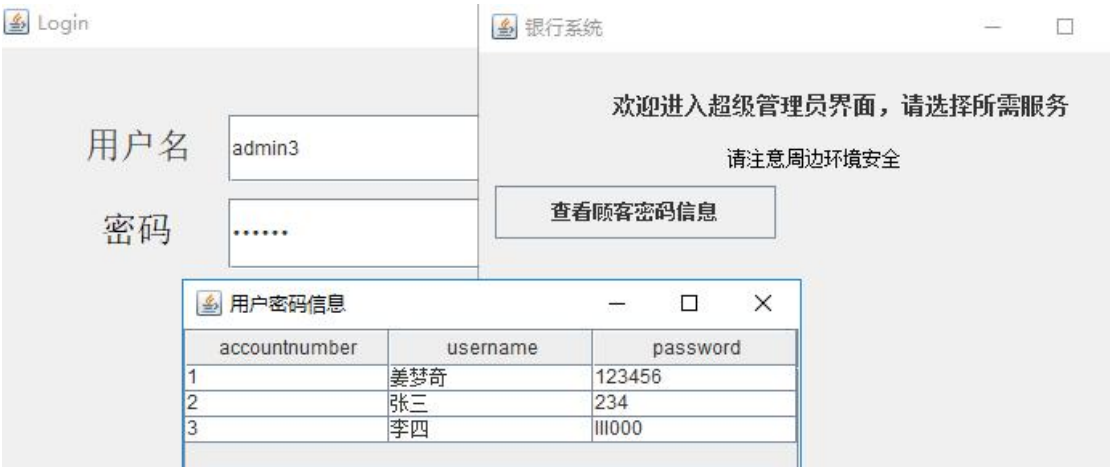
银行系统各用户的职责：





用户名	权限概述	关键字	备注
backup	对数据库进行备份	EVENT, LOCK TABLES, SELECT, SHOW DATABASES	EVENT 允许执行一次或周期性事件
monitor	监控数据库服务器，维护安全	EVENT, PROCESS, RELOAD, SHOW DATABASES, SHUTDOWN, SUPER	SUPER 允许终止任何查询，PROCESS 允许查看连接情况
security	创建、管理用户，设置权限，查看日志	CREATE USER, GRANT OPTION, RELOAD, SHOW DATABASES, SELECT	GRANT 只允许 传递本身有的权限





以上是该银行系统中，各部分的责任划分；

责任分离原则的定义是：一个交易至少两个人来完成；

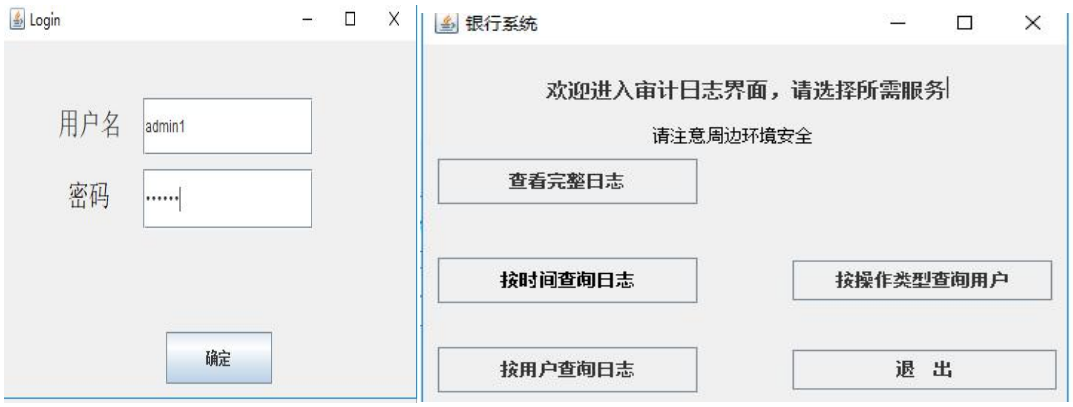
1. 无论存钱、取钱、转账、开户，这是一个银行职员与用户的交互的过程，责任对象设计银行与用户，满足责任分离原则；

2. 银行管理人员中，在存钱的过程中，有人专门负责更新顾客表中的信息，有人专门负责把交易记录写入日志中，满足责任分离原则；

3. 当用户投诉到，发现存款少了，先到柜台和工作人员确认，确实少了，此时，柜台工作人员报告管理员查看数据库中的完整的顾客信息表，查询余额，发现确实少了，此时，报告审计人员，查看日志，确认存款去向，如果发现是银行内部系统的问题，此时立即报告后端人员，进行系统修复，并修正用户信息表中的信息，根据日志记录，返回出问题之前的状态；这个过程设计多个对象，满足责任分离原则。

3.5 保存审计日志。

在前端，管理员 admin1 是可以查看日志，并对日志进行查询：



点击查看完整日志，得到如下表格：

日志信息

hid	accountnumber	money	content	time	worker
1	1	99.9	开户	2018-12-26 :07:19:20	aaa
2	2	100.0	开户	2018-12-26 :07:19:50	aaa
3	3	66.6	开户	2018-12-26 :07:20:11	aaa
4	1	1000.0	存款	2018-12-26 :07:38:45	bbb
5	2	2000.0	转账给账号3	2018-12-26 :07:39:32	aaa
6	3	66.6	取款	2018-12-26 :09:34:59	aaa
7	3	66.6	取款	2018-12-26 :09:35:06	aaa
8	3	66.6	取款	2018-12-26 :09:35:28	aaa
9	1	11.1	取款	2018-12-26 :09:37:32	bbb
10	3	11.1	取款	2018-12-26 :09:37:43	aaa
11	2	11.1	取款	2018-12-26 :09:38:06	bbb
12	4	2000.0	开户	2018-12-27 :09:31:03	aaa
13	1	500.0	转账给账号3	2018-12-27 :09:33:16	bbb
14	3	500.0	收到转账来自账户1	2018-12-27 :09:33:16	bbb

也可以通过账户查询操作：

结果为：

日志信息

hid	accountnumber	money	content	time	work
1	1	99.9	开户	2018-12-26 :07:19:20	aaa
4	1	1000.0	存款	2018-12-26 :07:38:45	bbb
9	1	11.1	取款	2018-12-26 :09:37:32	bbb
13	1	500.0	转账给账号3	2018-12-27 :09:33:16	bbb

后端也可以查看日志信息：

```
mysql> select * from t_history;
+-----+---------------+-------+-------------------------+---------------------+--------+
| hid | accountnumber | money | content                 | time                | worker |
+-----+---------------+-------+-------------------------+---------------------+--------+
|   1 |             1 |  99.9 | 开户                    | 2018-12-26 :07:19:20 | aaa    |
|   2 |             2 |   100 | 开户                    | 2018-12-26 :07:19:50 | aaa    |
|   3 |             3 |  66.6 | 开户                    | 2018-12-26 :07:20:11 | aaa    |
|   4 |             1 |  1000 | 存款                    | 2018-12-26 :07:38:45 | bbb    |
|   5 |             2 |  2000 | 转账給账号3             | 2018-12-26 :07:39:32 | aaa    |
|   6 |             3 |  66.6 | 取款                    | 2018-12-26 :09:34:59 | aaa    |
|   7 |             3 |  66.6 | 取款                    | 2018-12-26 :09:35:06 | aaa    |
|   8 |             3 |  66.6 | 取款                    | 2018-12-26 :09:35:28 | aaa    |
|   9 |             1 |  11.1 | 取款                    | 2018-12-26 :09:37:32 | bbb    |
|  10 |             3 |  11.1 | 取款                    | 2018-12-26 :09:37:43 | aaa    |
|  11 |             2 |  11.1 | 取款                    | 2018-12-26 :09:38:06 | bbb    |
|  12 |             4 |  2000 | 开户                    | 2018-12-27 :09:31:03 | aaa    |
|  13 |             1 |   500 | 转账給账号3             | 2018-12-27 :09:33:16 | bbb    |
|  14 |             3 |   500 | 收到转账来自账户1       | 2018-12-27 :09:33:16 | bbb    |
|  15 |             1 |    90 | 存款                    | 2018-12-27 :10:19:29 | aaa    |
+-----+---------------+-------+-------------------------+---------------------+--------+
```

3.6 遵循 Clark-Wilson 模型，定义应用系统的完整性限制条件

1.系统中，执行操作之前、之后，系统一直处于一致性状态

比如：用户的余额信息；

操作之前的用户余额加上日志中记录的存款信息，应该等于操作之后的用户余额；

操作之前的用户余额减去日志中记录的取款信息，应该等于操作之后的用户余额；

2.模型考虑如下几点：

1）主体必须被识别和认证

在本系统中主体是银行管理员，每一个管理员的身份都在系统中有记录，管理员在进行相关操作之前要进行身份认证登陆系统；

2）客体只能通过规定的程序进行操作

本系统中客体指用户，用户只有通过柜台普通人员运行程序，才能进行开户、存款、取款等操作。

3）主体只能执行规定的程序

对于用户管理员来说，每个人都有规定的权限，只能做职责之内的事，比如 admin1 只能查看日志；

4）必须维护正确的审计日志

日志系统前端后端都有对应人员在维护，可以确保日志正确；

5）系统必须被证明能够正确工作

系统可以正常运行，所以系统能被证明正确；

3.7 遵循 Clark-Wilson 模型的证明规则和实施规则，并在设计报告中有所体现。

实体：

CDI：受限数据项

数据要受完整性约束，本系统中所有数据都是受限数据项，在操作前后均保持一致；

UDI：非受限的数据项

数据不受完整性约束

TP: 交易过程

将系统从一个有效状态转移到另一个有效状态的过程，本系统的交易过程包括：开户、存款、取款、转账。

IVP: 完整性验证过程

检查 CDI 遵守完整性限制的过程

实施规则 1：

TP-->CDI ：保证具有关联关系

每一个受限数据项在进行交易操作之前，交易过程 TP 都会被验证；

每一个受限数据项在进行交易操作之后，都会被验证完整性；

实施规则 2：

user--TP-->CDI ：关联关系

TP 操作 CDI 时，保证操作用户有权对相应 CDI 做操作，TP 所代表的用户是 CDI 的真实用户；在本系统中，顾客通过柜台工作人员进行相应的 TP 操作，此过程中,工作人员有权对顾客表中该顾客的余额等信息进行修改,并写进审计日志中。

实施规则 3：

验证用户身份

无论是普通人员给顾客进行操作，还是管理员进入后台操作，都要输入用户密码验证身份；

证明规则 3：

满足责任分离原则

责任分离原则已经在上文讲解过；

实施规则 4：

谁可以执行 TP,需被授权；

只有银行普通员工可以与用户进行交互，执行 TP 操作。

四．实验收获

通过本次实验，首先对数据库的知识有了接触与学习，目前对简单的数据库操作能很好地掌握，比如建数据库，建用户，建表，设置权限等；

数据库编程上，主要是 java 的运用，eclispe 与数据库的连接，实现对数据库的读写，同时，感受到，数据库的使用使程序更简洁，更方便；比用文件要简单；在用 java 编程的时候，图形化界面的书写，让我对 java 里 swing 的掌握更熟练，能更好的写出好看、简洁的 GUI 界面；

Clark-Wilson 模型的使用，让我对责任分离原则、完整性、一致性等有了更好的了解，同时，设计一个系统时候，应该更加注意系统的安全性，重视职责分配，权限设置。

最后，本次银行系统的设计，对银行内部各人员职责有了更清楚的理解。