

**计算机系统安全实验报告**

**实验四：完整性访问控制系统设计与实现**

学　　 号 1190201308

班　　 级 1903202

学 生 陈东鑫

日 期 2021年12月15日

**一、实验环境：**

操作系统 Win10

数据库 MySQL 8.0.26

编译器 IntelliJ IDEA 2021.2

语言 Java 8.0.291

**二、系统安全策略和设计：**

（1）给出应用系统的安全策略文档。

系统身份分为四种：**普通用户**、**管理员**、**主管、审计人员**，其中普通用户和管理员具有系统执行权限，而主管只有授权和撤权权限，不能进入系统执行用户和管理员操作。

普通用户：**查询余额** 和 **提出存款/取款申请**

管理员：**查询所有普通用户余额** 和 **处理存款/取款申请**

主管：**将普通用户晋升为管理员** 或 **将管理员降级为普通用户**；在更改权限时将其余额归零或初始化为0，放置用户被提升为管理员后恶意调换自身余额。

审计人员：查看审计日志

操作系统：

普通用户：只能**提出存款/取款申请**，不能**处理存款/取款申请。**

管理员：只能**处理存款/取款申请**，不能**提出存款/取款申请。**

身份授权：

主管：只能授权和撤权，由于权限过大，不能分离，故不能操作系统。

审计：

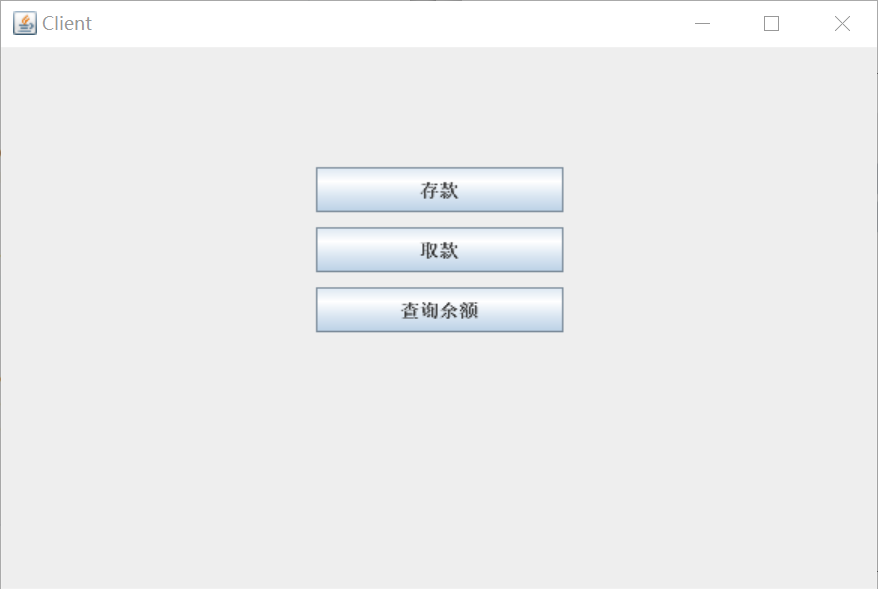
审计人员：查看审计日志。

（2）提供交互界面，能够完成录入、查询等功能。

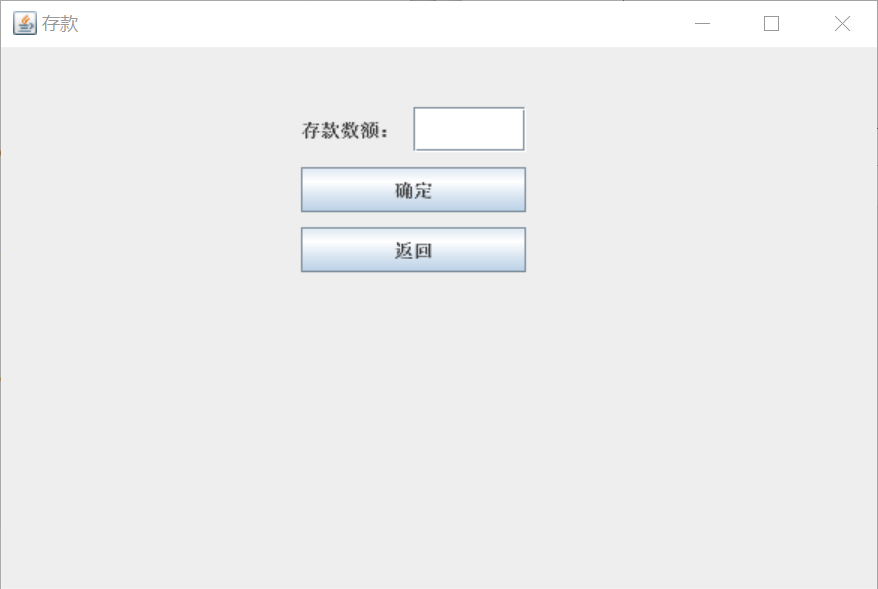
①登陆界面：可选定四种登录角色，当使用用户名登录错误的用户时，会提示登录错误。如使用主管身份的用户名以管理员身份登录，试图去篡改用户的余额，但系统不能使其成功登录。



②Client界面，普通用户身份登录：初始菜单，可以在菜单使用三种功能如下图按钮所示。



存取款界面：需要输入存取款数额，点击确定后，用户提出存取款请求；点击返回回到菜单。

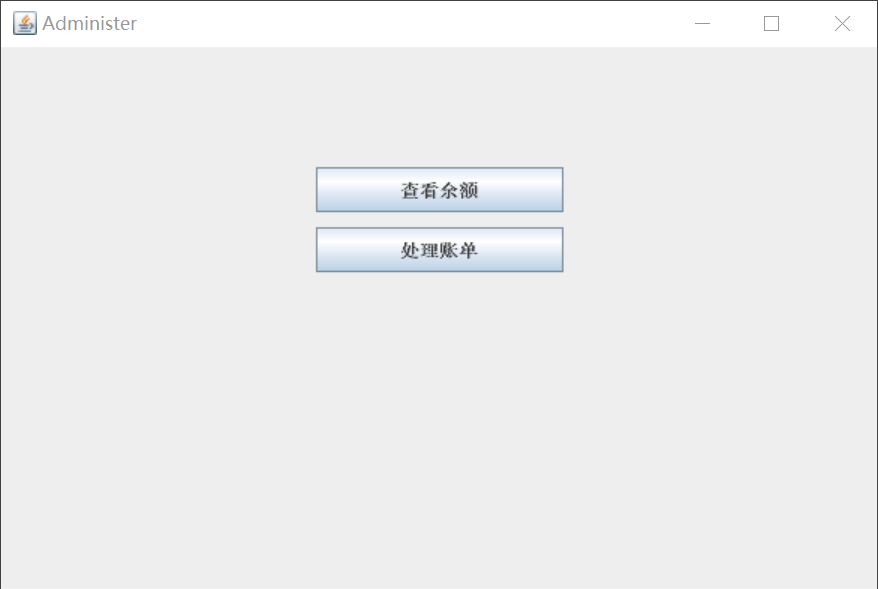




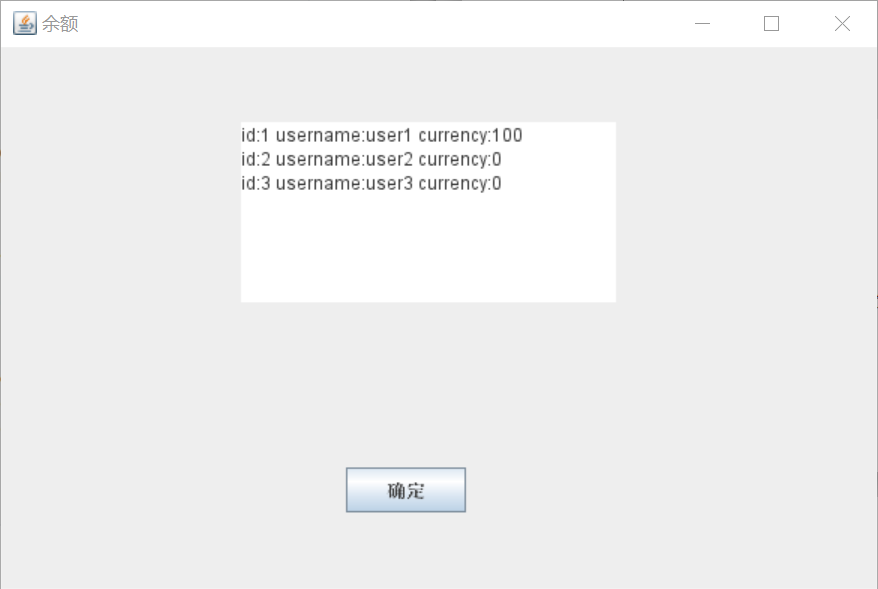
查询余额界面：可以看到自己的余额。



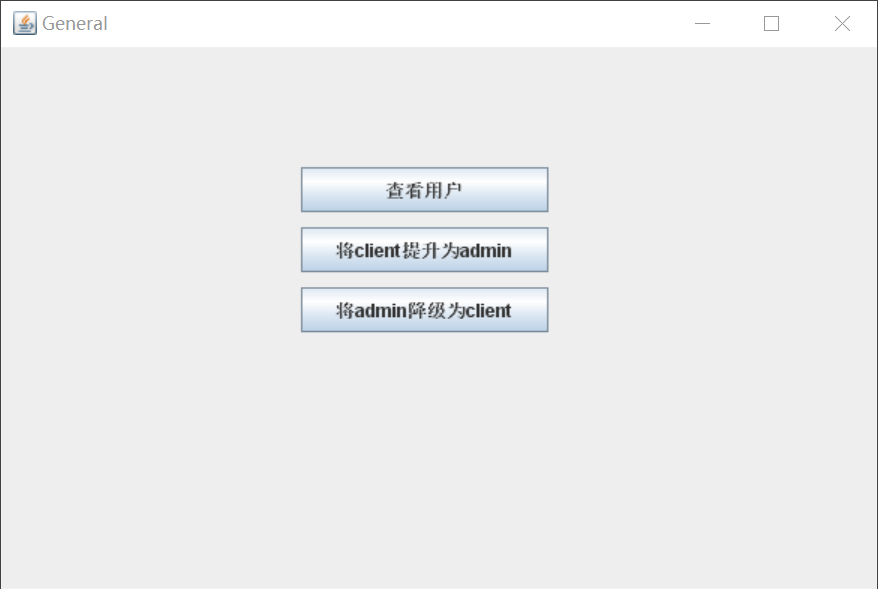
③Administer界面，管理员身份登录：初始菜单，可以使用两种功能，如下图按钮所示。



查看所有用户余额界面：查看所有普通用户余额。



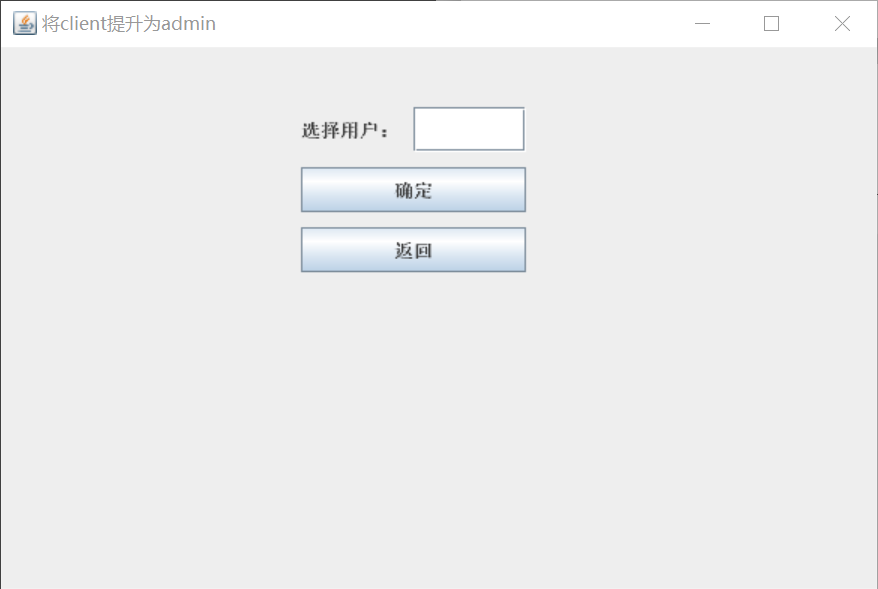
④General界面，主管身份登录：初始菜单，可以使用三种功能，如下图按钮所示。



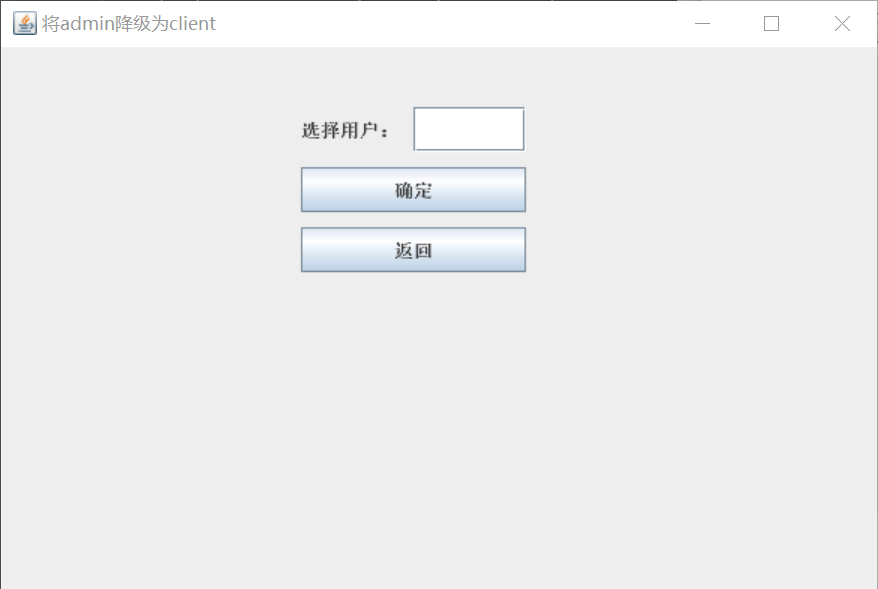
查看所有用户身份界面：包括普通用户和管理员，方便授权和撤销权限以改变用户身份。



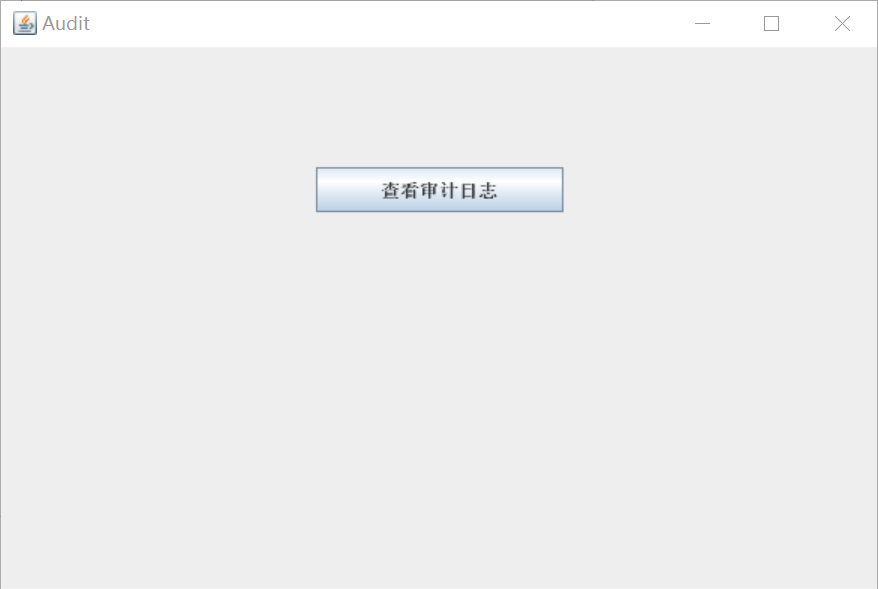
将client提升为admin界面：通过撤销原client权限，授权admin权限，将普通用户身份变为管理员，在改变时需要修改client和admin的表，同时将被提升的client的余额改为0，强制其将余款提取，防止其篡改自身的余额再降级为client后完整性出错。



将admin降级为client界面：通过撤销原admin权限，授权client权限，将管理员身份变为普通用户，在改变时需要修改client和admin的表，同时将被降级的admin的余额改为0，将其作为管理员时的余款（作为管理员时余款表中的余款字段无效）置为零，防止篡改自身的余额再降级为client后完整性出错。



⑤Audit界面，审计人员身份登录：



⑥工作流程

普通用户提出存款/取款

管理员处理存款/取款请求



管理员查看余额可以看到成功修改余额

账单处理完毕后，提示没有账单

（3）满足责任分离、功能分离原则。

责任分离：存款/取款需要 普通用户 和 管理员 两人完成

①普通用户 提出存款/取款请求。

②管理员 处理存款/取款请求，完成存款/取款对应的数据库记录修改。

没有普通用户，即没有存款/取款请求，管理员不能凭空随意修改余额表中数据；没有管理员，普通用户不能自己独自随意修改余额表中数据；存款/取款需要由普通用户和管理员两人共同完成。

（4）保存审计日志。

在用户登录、操作数据库记录时，需要将操作用户身份、用户名、操作结果记录到审计日志log.txt。

编写Log类，在程序开始时，初始化向日志文件log.txt的输出流。

try {  
 File file = new File("log.txt");  
 FileOutputStream output = new FileOutputStream(file, true);  
 Log.*output* = output;  
} catch (IOException e) {  
 System.*out*.println("open log.txt failed");  
}

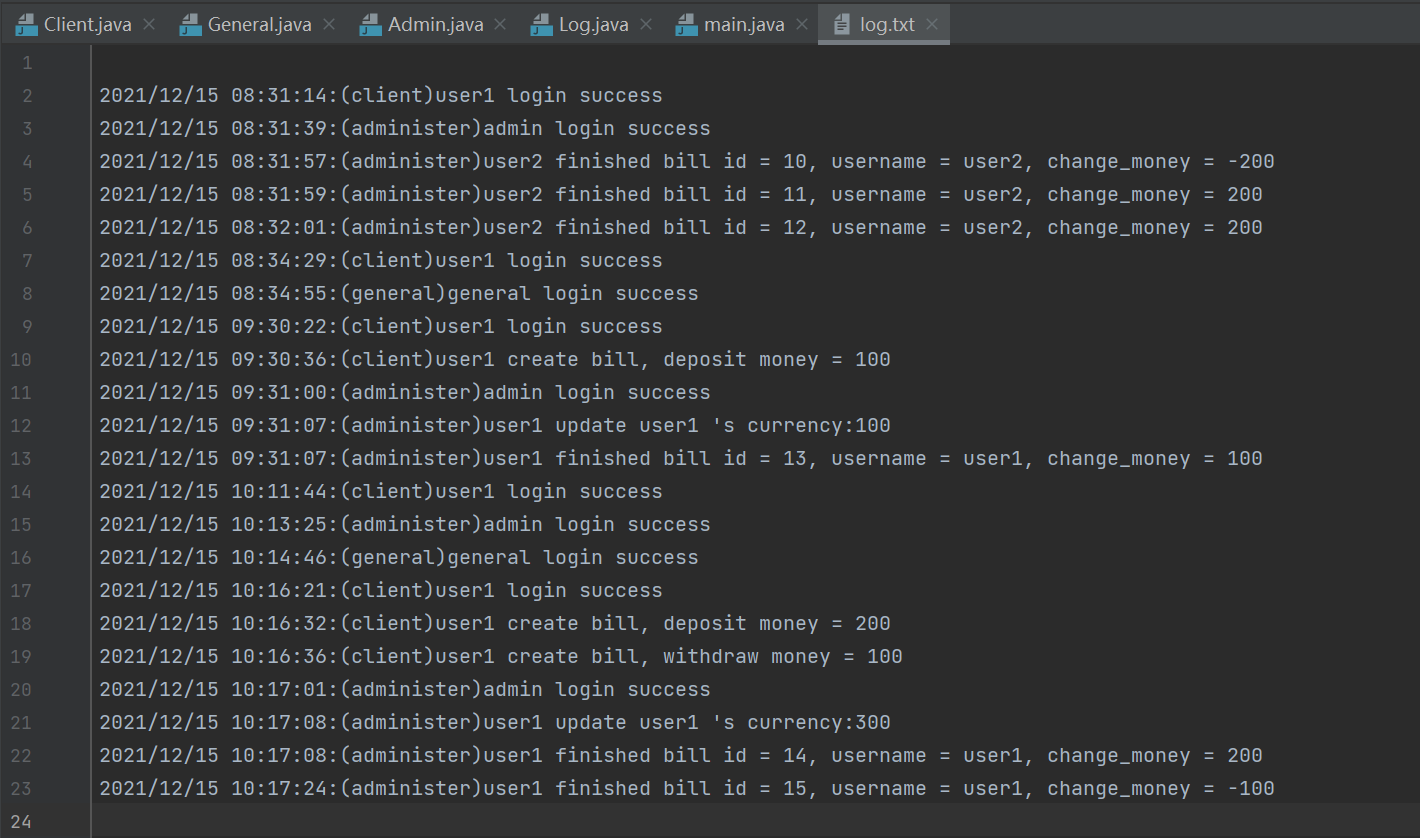
得到当前时间:

Date now = new Date();  
SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");  
String current\_time = dateFormat.format(now);

将输出流作为static成员变量，在写入日志时直接调用类的输出流进行写入。

try {  
 Log.*output*.write((current\_time + ":" + username + " delete bill id = " + bill\_id + ", username = " + username + ", change\_money = " + change\_money + "\n").getBytes(StandardCharsets.*UTF\_8*));  
} catch (IOException e) {  
 System.*out*.println("写入失败");  
}

查看log.txt文件：



（5）遵循Clark-Wilson模型，定义应用系统的完整性限制条件。

Clark-Wilson模型考虑如下⼏点：

1. 主体必须被识别和认证

任何用户登陆时进行身份认证，在所选登录身份与用户真实身份不匹配时登录失败。故主体在系统中进行的任何操作都是被识别和认证的。

2. 客体只能通过规定的程序进⾏操作

客体为普通用户的存款余额、存款/取款请求和普通用户有效性标志，对于存款余额只有通过普通用户请求和管理员批准时才能更改；对于存款/取款请求，只有用户在请求存款/取款时能插入新的请求，只有管理员处理存款/取款申请时能查看并删除；对于有效性标志，在主管进行权限授予和撤销时进行更改。故客体只能通过规定的程序进行操作。

3. 主体只能执⾏规定的程序

三种主体能执行自己规定的程序并且不重合。普通用户查看自身余额和提出存款/取款申请，管理员查看所有人余额和处理存款/取款申请，主管查看所有用户和授予/撤销权限。各用户功能补充和，能够实现责任分离。

4. 必须维护正确的审计⽇志

在用户登录、操作数据库记录时，需要将操作用户身份、用户名、操作结果记录到审计日志Log.txt。

5. 系统必须被证明能够正确⼯作

系统通过各种保障，能够正确工作不出错。

（6）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。

CR1：当任意 IVP 运⾏时，它必须保证所有的CDI处于有效状态

体现：普通用户想要取款/存款时，先要提出申请，在管理员处理存款/取款申请；当CDI无效时，即收到不再普通用户表里的用户的申请（可能为晋升为管理员的原用户的申请），管理员可以查询用户表和用户表中的有效标志检查到CDI无效；并且即使管理员的继续操作也无效，因为一方面管理员没有权限而无法创建新用户，另一方面有效值为false的用户余额是无效的。

CR2：对相关联的CDI，⼀个TP必须将这些CDI从⼀个有效状态转到另⼀个有效状态

体现：存款/取款时，请求和余额的更改为关联的CDI，想要完成这一TP，必须由用户向账单表插入请求，由管理员从账单表得到请求并更改对应余额，这些CDI在这两步操作中保持对应并且有效。

ER1：系统要维护关联关系，保证经过验证的TP操作相应的CDI

体现：系统各个身份的功能和职责是分离的，一个TP的不同操作由不同的用户负责，并且这些用户都是被识别的，不可能出现A身份执行B身份功能的情况。由Clark-Wilson模型，客体只能通过规定的程序进⾏操作，对于存款余额只有通过普通用户请求和管理员批准时才能更改；对于存款/取款请求，只有用户在请求存款/取款时能插入新的请求，只有管理员处理存款/取款申请时能查看并删除；对于有效性标志，在主管进行权限授予和撤销时进行更改。故客体只能通过规定的程序进行操作。

ER2：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

体现：对于存款余额只有通过普通用户请求和管理员批准时才能更改；对于存款/取款请求，只有用户在请求存款/取款时能插入新的请求，只有管理员处理存款/取款申请时能查看并删除；对于有效性标志，在主管进行权限授予和撤销时进行更改。故客体只能通过规定的程序进行操作。即只有有权限的身份的用户，才能使用该身份在系统中拥有的功能，去操作对应的CDI。

CR3：系统执⾏操作时，符合责任分离原则

体现：普通用户和管理员的功能互相排斥，需要二者配合才能完成一个存款/取款功能，在一个TP中，两用户的责任分明，在出现错误时能够追究责任。

ER3：系统执⾏TP前，应验证⽤户身份

体现：用户在登录之前需要选择用户登录进行登录，并验证用户身份，使用不匹配的身份登录会导致错误。

ER4：只有可以授予TP访问规则的主体才能修改列表中相应的表项，授权主体不能执⾏TP操作

体现：授权主体为主管，但主管无法进入系统进行操作，只能进行授权和撤权工作。即授权主体不能执行TP操作。

只有被主管授予权限的普通用户和管理员才能配合进行存款/取款操作。

**三、数据库建立：**

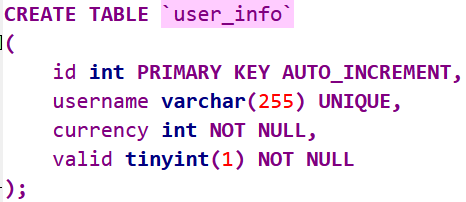
详见数据库脚本文件db.sql。

建立四个表，分别为user\_info、admin\_info、bill\_info、audit\_info

（1）user\_info，普通用户表

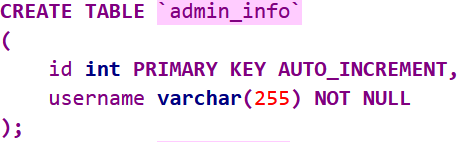
表中字段为用户ID、用户名、余额、有效性

当用户被转为管理员时，有效性变为false，将其唯一ID转到admin\_info表，并将其余额归零，防止其转为管理员后利用管理员权限刷钱。



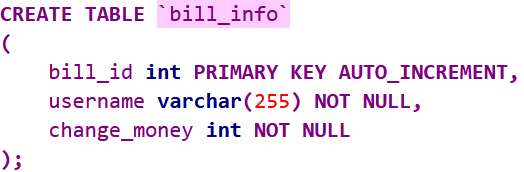
（2）admin\_info，管理员表

表中字段为管理员ID（继承自用户表ID）、用户名。在管理员降级为用户时余额仍要归零，防止其转为管理员后利用管理员权限刷钱。



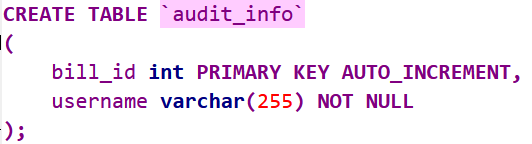
（3）bill\_info，账单表

表中字段为账单ID、存款/取款申请人用户名、存款/取款数额（取款为负数，存款为正数）

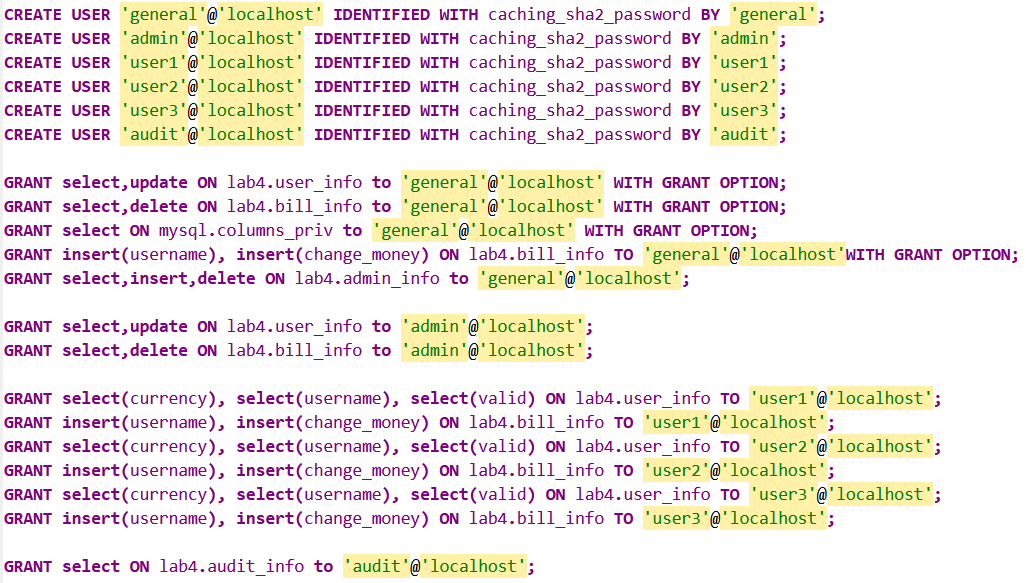


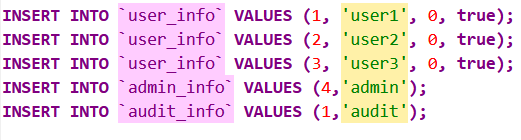
（3）audit\_info，审计人员表

只有审计人员可以访问



创建用户并授予权限





其中主管具有普通用户和管理员权限并有授予/撤销这些权限的权限。

**四、源代码：**

详见附件。基本模块实现代码与实验三相同。

**五、心得体会：**

在实验三的基础上，完善了系统，设计了满足银行商业公司的完整性访问控制需求的完整性访问控制系统。了针对公司业务需求的应用系统安全策略。安全策略中要明确指明对公司的要求与约束,和对客户的要求与约束,区分各自的责任。为商业公司设计系统，应用系统满足完整性需求。需求中包含责任分离、功能分离、审计。