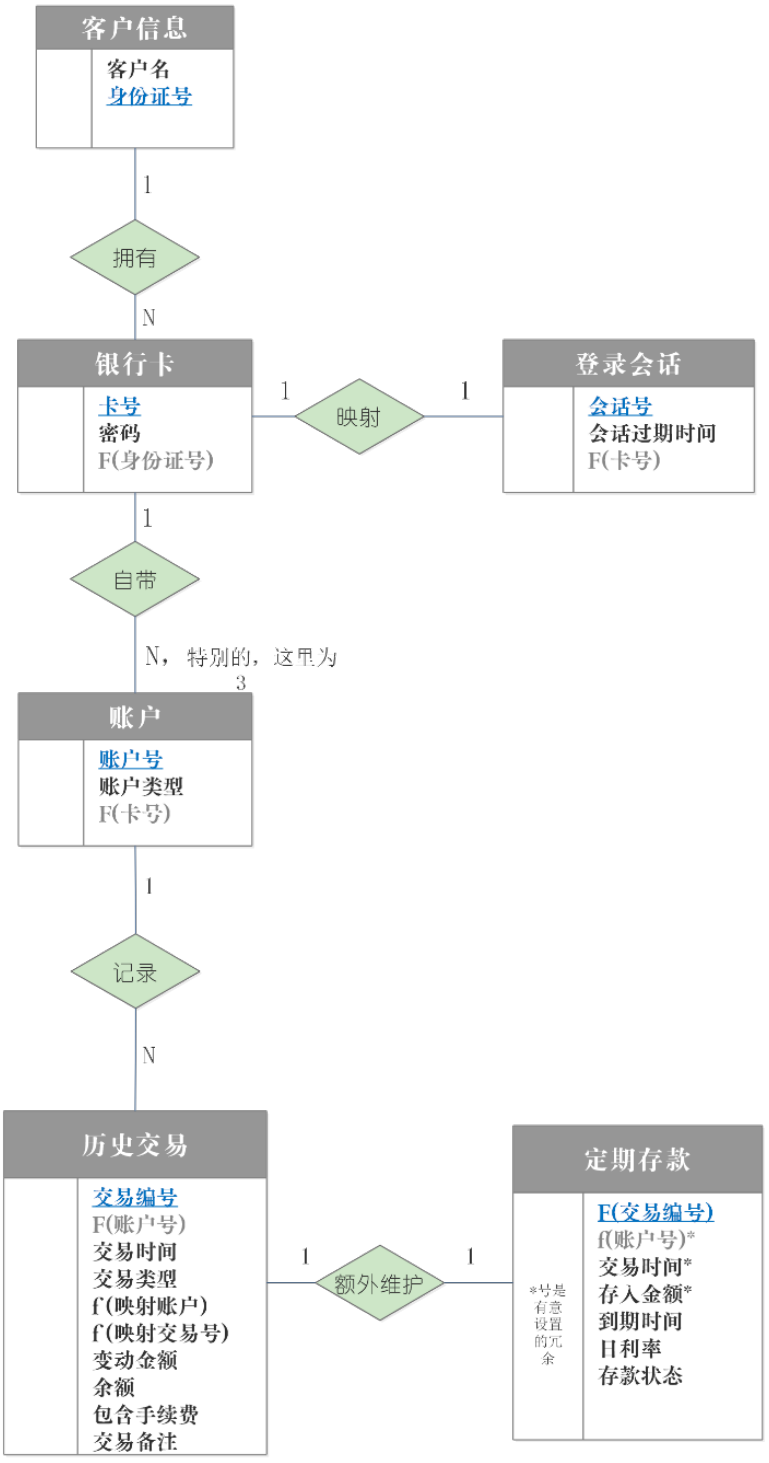
数据库课程设计作业

ATM 系统（数据库实现部分）设计报告

一、基本表的建立与完整性约束

右图展示了经由逻辑结构设计讨论得到的带有具体的关系模式的细化 ER 图。



参照此图，建立名为 atm 的数据库，在库内建立 6 个基本表，各个表之间通过外

键联系：

1.1 客户信息表

```
CREATE TABLE `客户信息` (
    `客户名` TINYTEXT NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
    `身份证号` VARCHAR(20) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
    `客户状态` INT(10) NOT NULL DEFAULT '0',
    PRIMARY KEY (`身份证号`) USING BTREE
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB
;
```

客户信息表的核心包含客户名和身份证号

- 以**身份证号**作为主键；
- **客户名**为 TINYTEXT 类型，能够兼容较长的姓名。
- **客户状态**原先是设计用于记录客户信用相关信息的，但是更多的设计没有继续，就默认置 0，本项目中没有用到；

客户信息表独立于银行卡密表，这使得表的扩展性得到保证，未来若有增设客户信息字段(如地址，联系电话)的需要将会比较方便。

1.2 银行卡密表

```
CREATE TABLE `银行卡密` (
    `卡号` BIGINT(19) NOT NULL AUTO_INCREMENT,
    `密码` VARCHAR(50) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
    `身份证号` VARCHAR(20) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
    PRIMARY KEY (`卡号`) USING BTREE,
    INDEX `FK_银行卡密_客户信息` (`身份证号`) USING BTREE,
    CONSTRAINT `FK_银行卡密_客户信息` FOREIGN KEY (`身份证号`) REFERENCES `atm`.`客户信息` (`身份证号`) ON UPDATE NO ACTION ON DELETE NO ACTION
)
COLLATE='utf8mb4_0900_ai_ci'
ENGINE=InnoDB
AUTO_INCREMENT=1036
;
```

在项目设计上一名客户最多可持有 3 张银行卡，银行卡密基本表记录了相关信息：

- 该表以**卡号**为主键，卡号以自增 bigint 类型为默认值，确保了卡号不重复，实现了卡号的自动生成方法。
- 该表的**密码**虽然在实际前端要求用户输入六位数字密码，但是这个密码的传输存储具有一个加解密的过程，为了防止出错就设置为 char 类型了。
- **身份证号**作为与客户信息联系的外键。

1.3 临时会话表

```
CREATE TABLE `临时会话` (
    `会话号` BIGINT(19) NOT NULL,
```

```
    `卡号` BIGINT(19) NOT NULL,
    `会话过期时间` TIMESTAMP NOT NULL,
    `会话建立时间` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (`会话号`) USING BTREE,
    INDEX `FK_session_users` (`卡号`) USING BTREE,
    CONSTRAINT `FK_临时会话_银行卡密` FOREIGN KEY (`卡号`) REFERENCES `atm`.`银行卡
密` (`卡号`) ON UPDATE NO ACTION ON DELETE NO ACTION
)
COMMENT='会话记录表'
COLLATE='utf8_general_ci'
ENGINE=InnoDB
;
```

临时会话表用于建立 ATM 系统端登录后的会话与登录卡号的联系。

- 其中以**会话号**为主键；
- **卡号**为映射当前所操作的银行卡信息，作为外键与银行卡密表相联系；
- **会话过期时间**和**建立时间**将作为临时会话信息定时清理的依据。

1.4 卡内账户表

```
CREATE TABLE `卡内账户` (
    `账户号` VARCHAR(50) NOT NULL DEFAULT '' COLLATE 'utf8mb4_0900_ai_ci',
    `卡号` BIGINT(19) NOT NULL DEFAULT '0',
    `账户类型` VARCHAR(2) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
    PRIMARY KEY (`账户号`) USING BTREE,
    INDEX `卡号` (`卡号`) USING BTREE,
    CONSTRAINT `FK_卡内账户_银行卡密` FOREIGN KEY (`卡号`) REFERENCES `atm`.`银行卡
密` (`卡号`) ON UPDATE NO ACTION ON DELETE NO ACTION
)
COLLATE='utf8mb4_0900_ai_ci'
ENGINE=InnoDB
;
```

- 卡内账户表以**账户号**作为主键，该账户号由开户时的调用过程决定。而不选择自增序列，主要是希望开户过程生成与对应卡号相关联的前后缀或相关编码规则作为账户号，方便查验和关联。
- **卡号**作为外键构建了单张银行卡与卡内账户的联系，
- **账户类型**在本设计中主要包含“活期”，“定期”和“信用”三类，另外 ATM 系统账户设有一个“系统”账户类型。

1.5 账户流水表

```
CREATE TABLE `账户流水` (
    `交易编号` BIGINT(19) NOT NULL AUTO_INCREMENT,
    `交易时间` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `账户号` VARCHAR(50) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
    `交易类型` TINYTEXT NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
    `映射账户` VARCHAR(50) NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
    `映射交易号` BIGINT(19) NULL DEFAULT NULL,
    `变动金额` DECIMAL(20,2) NOT NULL DEFAULT '0.00',
    `余额` DECIMAL(20,2) NOT NULL DEFAULT '0.00',
    `包含手续费` DECIMAL(20,2) NOT NULL DEFAULT '0.00',
```

```
    `交易备注` TINYTEXT NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
    PRIMARY KEY (`交易编号`) USING BTREE,
    INDEX `FK_活期_users` (`账户号`) USING BTREE,
    INDEX `FK_活期_users_2` (`映射账户`) USING BTREE,
    INDEX `操作时间` (`交易时间`) USING BTREE,
    INDEX `FK_账户流水_账户流水` (`映射交易号`) USING BTREE,
    CONSTRAINT `FK_账户流水_卡内账户` FOREIGN KEY (`账户号`) REFERENCES `atm`.`卡内
账户` (`账户号`) ON UPDATE NO ACTION ON DELETE NO ACTION,
    CONSTRAINT `FK_账户流水_卡内账户_2` FOREIGN KEY (`映射账户`) REFERENCES `atm`.`
卡内账户` (`账户号`) ON UPDATE NO ACTION ON DELETE NO ACTION,
    CONSTRAINT `FK_账户流水_账户流水` FOREIGN KEY (`映射交易号`) REFERENCES `atm`.`
账户流水` (`交易编号`) ON UPDATE NO ACTION ON DELETE NO ACTION
)
COLLATE='utf8mb4_0900_ai_ci'
ENGINE=InnoDB
AUTO_INCREMENT=182
;
```

账户流水表是本设计中关系最为复杂的基本表，该表记录了不同客户之间所有账户的交易流水。

- 以自增序列的**交易编号**为主键，规避了以操作时间或（操作时间，账户号）为主键时，同时间（1 秒内）同账户操作带来的冲突。
- **交易时间**默认以执行插入操作时的系统当前时间，不需要特别指定；账户号作为外键与卡内账户表联系。
- **交易类型**用于简短记录此次交易的内容，如“存款”、“取款”、“转出”、“转入”等。
- **映射账户**表示与此次交易内容相关的操作对方或发起方账户号，在存取款时，映射账户为自身账户号；在转出操作时，映射账户号为转账到对方的账户号，在转入操作时，映射账户号为转账操作发起人对方的账户号；
- **映射交易号**反映了具有关联的交易对，例如转入和转出操作两条记录是成对进行的，因此两个操作的映射交易号互为彼此。映射交易号的设计目的在于有效的针对转账操作进行定位溯源，在本设计中暂未实际使用到；
- **变动金额**代表此次操作的涉及款项，取款，转出等操作的变动金额为负数，存款，转入等操作的变动金额为正数。该变动金额是包含手续费在内的金额。
- **余额**表征了该账户在该笔交易后账户的剩余可交易额度，在信用账户中，该值透支至-2000 元。余额只通过变动金额进行计算。
- **包含手续费**仅作为手续费的展示，不参与账户金额变动的计算。例如用户转出 500 元，手续费收取 1%，则操作记录显示变动金额 505.00，包含手续费 5.00，此时计算余额方式为最近一笔交易的余额减去 505.00，而不会计算为最近一笔交易的余额-505.00-5.00。
- **交易备注**反映了更为详细和必要的交易信息，如扣除手续费的原因等。

1.6 定期存款表

```
CREATE TABLE `定期存款` (
    `交易编号` BIGINT(19) NOT NULL AUTO_INCREMENT,
    `交易时间` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `账户号` VARCHAR(50) NOT NULL DEFAULT '0' COLLATE 'utf8mb4_0900_ai_ci',
    `存入金额` DECIMAL(20,2) NOT NULL DEFAULT '0.00',
```

```
    `到期时间` TIMESTAMP NOT NULL,
    `日利率` DECIMAL(20,6) NOT NULL DEFAULT '0.000000',
    `存款状态` INT(10) NOT NULL DEFAULT '0',
    PRIMARY KEY (`交易编号`) USING BTREE,
    INDEX `FK_定期存款_卡内账户` (`账户号`) USING BTREE,
    CONSTRAINT `FK_定期存款_卡内账户` FOREIGN KEY (`账户号`) REFERENCES `atm`.`卡内账户` (`账户号`) ON UPDATE NO ACTION ON DELETE NO ACTION,
    CONSTRAINT `FK_定期存款_账户流水` FOREIGN KEY (`交易编号`) REFERENCES `atm`.`账户流水` (`交易编号`) ON UPDATE NO ACTION ON DELETE NO ACTION
)
COLLATE='utf8mb4_0900_ai_ci'
ENGINE=InnoDB
AUTO_INCREMENT=160
;
```

定期存款是在账户流水的基础上额外维护的一个表，其建立的根本目的是适应业务需求，记录定期存款的**到期时间**，**利率**和**是否已取出（存款状态）**这三个信息，如果三个核心属性也写在账户流水表内会造成账户流水表的臃肿且利用率低。

定期存款表以账户流水表中涉及到定期存入的每条**交易号**为主键，并有意引入了如**交易时间**，**账户号**和**交易金额**这类冗余量，方便后台数据的查看校对，当然这些字段也可以进一步优化除去。

二、存储过程的编写

本项目后端没有显式的服务端封装，即不是传统的（客户端<->服务端<->数据库端）的模式，这带来了安全性相关的问题，如果在前端直接登录具有权限的账户执行 select,insert,update 等语句，那么如果前端软件遭到破坏后将会造成数据库管理权限的外泄，造成数据库入侵，带来威胁。

经过讨论，我们决定以数据库账户和权限为切入点，借由存储过程，模拟后端的接口封装。前端连接到数据库端的子账户仅能调用注册，登录，存取款等存储过程，实现更加细化的功能权限，而没有对任何基本表的通用增删改查权限。从而在一定程度上解决了软件被破坏数据库账号泄露等问题下的安全性问题。按照实验设计需求，数据库共设置有 8 个存储过程。

2.1 注册功能（signup）

```
delimiter $
CREATE DEFINER=`root`@`%` PROCEDURE `signup`(
    IN `name` TINYTEXT,
    IN `pwd` VARCHAR(50),
    IN `pid` BIGINT,
    OUT `id` INT,
    OUT `info` TEXT
)
BEGIN
    DECLARE num INT ;
    DECLARE acc TEXT;
    DECLARE ACCOUNT VARCHAR(50);
    SELECT COUNT(*) INTO num FROM 客户信息 WHERE 身份证号=pid;
    if num > 2 then
```

```
        SET id = -1;
        SET info = '开户失败，一位客户最多拥有三张银行卡';
    ELSE
        if num IS NULL OR num = 0 then
                INSERT INTO 客户信息(客户名,身份证号) VALUES (NAME,pid);
        END if;
        SELECT 客户名 INTO @kehuname FROM 客户信息 WHERE 身份证号=pid;
        if @kehuname = NAME then
            insert into 银行卡密(密码,身份证号) values(AES_DECRYPT(FROM_BASE64(pwd),
'HNUdatabase'),pid);
            select max(卡号) into id from 银行卡密 WHERE 身份证号=pid;
            SET acc = id+'';
            INSERT INTO 卡内账户(账户号,卡号,账户类型)VALUES(CONCAT('1-',acc),id,'活期');
            INSERT INTO 卡内账户(账户号,卡号,账户类型)VALUES(CONCAT('2-',acc),id,'定期');
            INSERT INTO 卡内账户(账户号,卡号,账户类型)VALUES(CONCAT('3-',acc),id,'信用');
            SELECT 账户号 INTO ACCOUNT FROM 卡内账户 WHERE 卡号=id AND 账户类型='活期';
            INSERT INTO 账户流水(账户号,交易类型,映射账户,变动金额,余额)VALUES(ACCOUNT,'开户
','ATM',0,0);
            SELECT 账户号 INTO ACCOUNT FROM 卡内账户 WHERE 卡号=id AND 账户类型='定期';
            INSERT INTO 账户流水(账户号,交易类型,映射账户,变动金额,余额)VALUES(ACCOUNT,'开户
','ATM',0,0);
            SELECT 账户号 INTO ACCOUNT FROM 卡内账户 WHERE 卡号=id AND 账户类型='信用';
            INSERT INTO 账户流水(账户号,交易类型,映射账户,变动金额,余额)VALUES(ACCOUNT,'开户
','ATM',0,0);
            SET info = '开户成功，请妥善保管您的卡号和密码，此卡为定活信用一卡通类型，同时带
有三个子账户';
        ELSE
            SET id = -1;
            SET info = '您输入的客户名与身份证信息不匹配，请重新输入。';
        END if;
    END if;
END $
```

注册存储过程传入参数为客户名 name，经过加密的客户密码 pwd，用户身份证号 tel；传出参数为注册结果卡号 id，提示信息 info。

首先在银行卡密表中统计该用户已拥有的银行卡数量，如果已经有三张卡了，就拒绝开户请求，返回的 id 为-1 以及 info 用以提示；

而后将新用户信息登记到客户信息表；

检查老用户发来的身份证号与客户名是否匹配，在匹配的情况下：

向银行卡密表中新增卡片信息，这个密码 pwd 是经过 AES 对称加密和 base64 编码的，因此存入数据库时要进行解码和解密：AES_DECRYPT(FROM_BASE64(pwd), 'HNUdatabase')，这里'HNUdatabase'就是 AES 的密钥，这里使用 EBC 加密方法，没有偏移量的设置在一些安全系统中 AES 密钥是动态变化的，本项目简化了这一过程只做一个静态的密钥作为展示。需要注意的是 BASE64 的编解码函数只有高版本的 MySQL 才自带，需要做好版本适配；

获取新增卡片的卡号,这里使用 max 函数获取自增主键中最大的一个；

向卡内账户新增三个账户号，分别对应活期，定期和信用卡账户，这里账户号的编码规则简单地设置成不同的数字前缀‘1-卡号’，‘2-卡号’，‘3-卡号’；

向账户流水信息新增三条流水，记录三个账户的开户信息。

返回卡号 id 和提示信息。

2.2 登录功能（login）

```
delimiter $
```

```
CREATE DEFINER=`root`@`%` PROCEDURE `login`(
    IN `id` BIGINT,
    IN `pwd` VARCHAR(50),
    OUT `state` INT,
    OUT `cookie` BIGINT
)
LANGUAGE SQL
NOT DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN
  DECLARE sessionid BIGINT DEFAULT (NOW()-20220000000000)*id;
  SELECT COUNT(*) INTO state FROM 银行卡密 WHERE 卡号=id AND 密码=
AES_DECRYPT(FROM_BASE64(pwd), 'HNUdatabase');
  if state = 1 then
    INSERT INTO atm.临时会话(卡号,会话号,会话过期时间) VALUES (id,sessionid,
TIMESTAMPADD(MINUTE,2,now()));
    SET cookie := sessionid;
  ELSE
   SET cookie := 0;
  END if;
  if state IS NULL then
    SET state=0;
    END if;
END
END $
```

登录功能的存储过程传入参数为卡号 id，加密后的密码 pwd，返回登录结果提示 state，会话号 cookie。

会话号的生成方法是将登录时的时间（月日时分秒）数字与卡号相乘。应该还有更加合理的生成方法，这里就不进一步研究了。

存储过程将通过 select 语句在银行卡密表中匹配与卡号和解密后密码相符的记录行。

若返回记录行为 1，说明卡密正确，将会话信息新增到临时会话表中，返回对应 cookie。会话信息需要手动存入卡号、会话号和会话过期时间，默认会话有效期为 120 秒，因此使用函数 TIMESTAMPADD(MINUTE,2,now())在当前时间的基础上增加两分钟。

这里需要注意的是该数据库部署于 linux 服务器，默认国际标准时间，时区 +00:00。因此需要调整到东八区，这里使用

```
set global time_zone = '+8:00';
set time_zone = '+8:00';
flush privileges;
```

修改全局时区与当前会话时区，并刷新配置。

2.3 改密功能（modify）

```
CREATE DEFINER=`root`@`%` PROCEDURE `modify`(
    IN `cookie` BIGINT,
    IN `oldpwd` VARCHAR(50),
    IN `newpwd` VARCHAR(50),
    OUT `state` int,
    OUT `info` TEXT
)
LANGUAGE SQL
NOT DETERMINISTIC
```

```
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN
    DECLARE id BIGINT;
    SELECT 卡号 INTO id FROM 临时会话 WHERE 会话号=cookie;
    if id IS NOT NULL then
    UPDATE 临时会话 SET 会话过期时间 = TIMESTAMPADD(MINUTE,2,now()) WHERE 会话号=cookie;
        SELECT COUNT(*) INTO state FROM 银行卡密 WHERE 卡号=id AND 密码=
AES_DECRYPT(FROM_BASE64(oldpwd), 'HNUdatabase');
        if state = 1 then
         UPDATE 银行卡密 SET 密码= AES_DECRYPT(FROM_BASE64(newpwd), 'HNUdatabase') WHERE 卡
号=id;
         SET info = '修改成功';
        ELSE
            SET info = '旧密码错误';
            SET state =0;
        END if;
    END if;
END
```

密码修改的存储过程传入参数为会话号 cookie，旧密码 oldpwd 和新密码 newpwd。

用户登录后的所有操作均需要传入会话号 cookie 用于安全验证与定位银行卡片。

存储过程通过 select 语句将 cookie 映射为卡号 id，并且如果 cookie 有效，说明该会话目前不是闲置状态，将重置会话过期时间。这两个操作在后面的所有存储过程中都有进行，之后不再赘述。

与登录类似的，通过 select 语句匹配旧密码以验证正确性，匹配正确后更新密码信息并传回相关提示。

2.4 存款功能（save）

```
CREATE DEFINER=`root`@`%` PROCEDURE `save`(
    IN `cookie` BIGINT,
    IN `ctype` VARCHAR(2),
    IN `money` DECIMAL(20,2),
    OUT `state` INT,
    OUT `info` TEXT,
    OUT `balance` DECIMAL(20,2)
)
LANGUAGE SQL
NOT DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN
    DECLARE id BIGINT;
    DECLARE acc VARCHAR(50);
    SELECT 卡号 INTO id FROM 临时会话 WHERE 会话号=cookie;
    if id IS NOT NULL then
    UPDATE 临时会话 SET 会话过期时间 = TIMESTAMPADD(minute,2,now()) WHERE 会话号=cookie;
    SELECT 账户号 INTO acc FROM 卡内账户 WHERE 卡号 = id AND 账户类型 = ctype;
    if ctype = '活期' or ctype = '信用' then
        SELECT 余额 INTO balance FROM 账户流水 WHERE 账户号 = acc  ORDER BY 交易时间 DESC
LIMIT 1;
        SET balance = balance+money;
```

```
        INSERT INTO 账户流水(账户号,交易类型,映射账户,变动金额,余额)VALUES   (acc,'存款
',acc,money,balance);
        SET state = 1;
        SET info = concat('存款成功！本次存入金额为：¥',money);
    ELSEIF ctype = '定期' then
        SELECT 余额 INTO balance FROM 账户流水 WHERE 账户号 = acc  ORDER BY 交易时间 DESC
LIMIT 1;
        SET balance = balance+money;
        INSERT INTO 账户流水(账户号,交易类型,映射账户,变动金额,余额)VALUES   (acc,'存款
',acc,money,balance);
        select 交易编号,交易时间 into @opid,@optime from 账户流水 where 账户号=acc and 交易
类型='存款' order by 交易时间 DESC LIMIT 1;
        INSERT INTO 定期存款(交易编号,交易时间,账户号,存入金额,到期时间,日利率)VALUES
(@opid,@optime,acc,money,TIMESTAMPADD(day,30,@optime),0.0001);
        SET state = 1;
        SET info = concat('存款成功！本次存入金额为：¥',money,',定期存款存期 1 个月(30 个自然
日)，月利率为 0.3%，仅支持整存整取（含利息）');
    END if;
    END if;
END
```

存款功能传入参数为会话号 cookie，用户选择的账户类型 ctype，存入金额 money，传出参数为交易成功的状态提示 state，提示信息 info。

通过卡号 id 与账户类型 cytpe 映射在卡内账户表内获得账户号 acc。

如果选择的账户是活期账户或者信用账户，那么只需要操作账户流水表即可。

获取账户流水表内该账户号下最新的的一条记录，将该记录的余额作为当前账户余额。

插入一条新的存款记录即可。交易类型为存款，映射账户为自身，变动金额即 money，新的余额即当前余额加上变动金额 money。

如果选择的账户是定期账户，那么在添加完账户流水记录后还要获取该记录的交易编号。

在定期存款表中记录该笔存款的信息，交易编号即为刚才获取的，到期时间在本项目中简化固定为 1 个月之后，同样通过 TIMESTAMPADD 函数生成对应时间戳，日利率默认为 0.0001，即月利率 0.3%，存款状态默认是 0，不需要显式设置。

2.5 取款功能（withdraw）

```
CREATE DEFINER=`root`@`%` PROCEDURE `withdraw`(
    IN `cookie` BIGINT,
    IN `ctype` VARCHAR(2),
    IN `money` DECIMAL(20,2),
    OUT `state` INT,
    OUT `info` TEXT,
    OUT `balance` DECIMAL(20,2)
)
LANGUAGE SQL
NOT DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN
    DECLARE id BIGINT;
    DECLARE acc VARCHAR(50);
    declare day_money DECIMAL(20,2);
    DECLARE month_times INT;
    SELECT 卡号 INTO id FROM 临时会话 WHERE 会话号=cookie;
```

```
    if id IS NOT NULL then
    UPDATE 临时会话 SET 会话过期时间 = TIMESTAMPADD(minute,2,now()) WHERE 会话号=cookie;
    SELECT 账户号 into acc FROM 卡内账户 WHERE 卡号=id AND 账户类型=ctype;
        if money >2000 then
            set state = -1;
            SET info = '单次取款金额不得超过 2000 元';
        ELSE
            SELECT SUM(变动金额) into day_money FROM 账户流水 WHERE TO_DAYS(交易时间) = TO_DAYS(NOW())
AND 账户号 = acc AND  交易类型='取款';
            if day_money IS NULL then
                SET day_money = 0;
            END if;
            if day_money-money<-5000 then
                set state = -2;
                SET info = '单日累计取款金额不得超过 5000 元';
            else
              SELECT 余额 INTO balance FROM 账户流水 WHERE 账户号 = acc ORDER BY 交易时间 DESC LIMIT 1;
                if ctype = '活期' then
SELECT COUNT(*) into month_times FROM 账户流水 WHERE DATE_FORMAT(交易时间, '%Y%m') = DATE_FORMAT(NOW(),
'%Y%m') AND 账户号 = acc AND  交易类型='取款';
                    if month_times >= 5 then
                        if balance-money-2>0 then
                          INSERT INTO 账户流水(账户号,交易类型,映射账户,变动金额,余额,包含手续费)VALUES
                                (acc,'取款',acc,0-money-2,balance-money-2,-2);
                            set state = 2;
                            SET info = '取款成功，本月累计取款金额超过 5 次，收取手续费 2 元';
                        ELSE
                            set state = -2;
                            SET info = '您的余额不足以扣除取款金额与手续费之和';
                        END if;
                    ELSE
                        if balance -money > 0 then
                            INSERT INTO 账户流水(账户号,交易类型,映射账户,变动金额,余额)VALUES
                                (acc,'取款',acc,0-money,balance-money);
                            set state = 1;
                            SET info = '取款成功';
                        ELSE
                            set state = -2;
                            SET info = '您的余额不足以扣除取款金额';
                        END if;
                    END if;
                ELSEif ctype = '定期' then
 SELECT COUNT(*) into @match_op from 定期存款 WHERE 存款状态=0 AND 存入金额=money AND 账户号=acc;
                    if @match_op is null or @match_op = 0 then
                        set state = -2;
                        SET info = '您的账户中没有与取款金额匹配的未取出历史存款';
                    else
                        SELECT COUNT(*) into @match_op from 定期存款 WHERE 存款状态=0 AND 存入金额
=money and 到期时间<NOW() AND 账户号=acc;
                        if @match_op is null or @match_op = 0 then
             select 交易编号 into @match_op from 定期存款 WHERE 存款状态=0 AND 存入金额=money AND 账户
号=acc order by 到期时间 desc limit 1;
             select 账户号 into @huoqi_id from 卡内账户 where 卡号=id and 账户类型='活期';
                            SET @dingqib = balance;
SELECT 余额 INTO balance FROM 账户流水 WHERE 账户号 = @huoqi_id ORDER BY 交易时间 DESC LIMIT 1;
                            if balance <10 then
                                set state = -2;
SET info = '活期账户余额不足！您的该笔存款尚未到期，取出存款需要从活期账户扣除 10 元手续费';
                            else
                                UPDATE 定期存款 set 存款状态 = 2 where 交易编号=@match_op;
                                insert into 账户流水(账户号,交易类型,映射账户,变动金额,余额,交易备
注)VALUES (acc,'取款',acc,0-money,@dingqib-money,"取出未到期存款，于活期账户扣除手续费 10 元");
select max(交易编号) into @src_opid from 账户流水 where 账户号=acc and 交易类型='取款';
                                insert into 账户流水(账户号,交易类型,映射账户,映射交易号,变动金额,余额,
包含手续费,交易备注)VALUES (@huoqi_id,'代扣',acc,@src_opid,-10,balance-10,-10,"定期账户取出未到期存款，
扣除手续费 10 元");
                                set state = 1;
                            SET info = '取款成功，该笔存款尚未到期，已从活期账户中代扣 10 元手续费。';
                            end if;
```

```
                    else
        SELECT 余额 INTO @dingqib FROM 账户流水 WHERE 账户号 = acc ORDER BY 交易时间 DESC LIMIT 1;
        SELECT 余额 INTO balance FROM 账户流水 WHERE 账户号 = @huoqi_id ORDER BY 交易时间 DESC LIMIT 1;
        select 交易编号,日利率 into @match_op,@lilv from 定期存款 WHERE 存款状态=0 AND 存入金额=money
AND 账户号=acc order by 到期时间 asc limit 1;
                        UPDATE 定期存款 set 存款状态 = 1 where 交易编号=@match_op;
                        insert into 账户流水(账户号,交易类型,映射账户,变动金额,余额,交易备注)VALUES
(acc,'取款',acc,0-money,@dingqib-money,"取出到期存款，利息另转至活期账户");
        select 账户号 into @huoqi_id from 卡内账户 where 卡号=id and 账户类型='活期';
        select max(交易编号) into @src_opid from 账户流水 where 账户号=acc and 交易类型='取款';
                        insert into 账户流水(账户号,交易类型,映射账户,映射交易号,变动金额,余
额)VALUES (@huoqi_id,'定期利息',acc,@src_opid,@lilv*30*money,balance+@lilv*30*money);
                        set state = 1;
                        SET info = '取款成功,存款利息已取出至活期账户。';
                    end if;
                end if;
            ELSEif ctype = '信用' then
                if balance -money< -2000 then
                    set state = -2;
                    SET info = '取出金额超出可透支额度';
                else
                    INSERT INTO 账户流水(账户号,交易类型,映射账户,变动金额,余额)VALUES
                                (acc,'取款',acc,0-money,balance-money);
                    set state = 1;
                    SET info = '取款成功';
                END if;
                END if;
            END if;
        END if;
    END if;
    SELECT 余额 INTO balance FROM 账户流水 WHERE 账户号 = acc ORDER BY 交易时间 DESC LIMIT 1;
END
```

取款功能和转账功能在引入定期账户的情况下是最为复杂的两个存储过程。

取款存储过程的传入参数和传出参数与存款一致。

首先进行单次取款限额判断以及单日取款限额判断，前者直接将传入的取款金额与 2000 比较，超过 2000 则拒绝取款；后者在账户流水表内以条件 TO_DAYS(交易时间) = TO_DAYS(NOW())以及其他条件定位当日取款记录，通过 sum 聚集函数对操作类型为“取款”，账户号为该用户账户号的交易记录累加变动金额，如果累加和小于-5000（取款的变动金额总为负数），就拒绝取款。

通过账户号 acc 找到最新的一条交易记录作为旧余额。

按照账户类型的不同分支处理：

1）在活期账户下，借助限制条件 DATE_FORMAT(交易时间, '%Y%m') = DATE_FORMAT(NOW(), '%Y%m')定位到本月的交易记录，利用 count 聚集函数进行统计，如果本月进行过取款交易满 5 次，即本次是第 6 次取款，那么本次取款将计划扣除 2 元手续费。

计算余额是否足以扣除此次取款金额以及手续费（若含）。若不足以扣除，则返回失败信息；若可以正常取款，则在账户流水表中插入新的交易记录。返回取款成功的信息。

2）在定期账户下，只支持整存整取，所有手续费和利息由活期账户代扣或接收。

首先用 count 在定期存款表内统计与此次取款请求的金额相符的历史存款，例如取款金额为 500，就在表内统计存入金额为 500，存款状态为 0（未取出），账户号为 acc 的记录行数量。如果查询结果为空，就返回取款失败。

若查到了可以取款的存款记录，就分两种情况处理：

通过条件“到期时间<NOW()”筛选已经到期的存款，如果结果为空，说明就需要取出未到期的存款。

在策略上我们通过交易时间降序选择最近的一笔未到期存款，目的是让其他存款尽可能存得久一些。取出这笔交易的交易号。

未到期存款需要计划从活期账户中扣除手续费 10 元。这又需要进行活期余额是否充足的判断。在卡内账户表中定位该银行卡的对应活期账户号。在账户流水表内查询该活期账户的余额，判断余额是否大于 10。如果不够，就返回交易失败信息。

活期余额充足时，可以正式进行取款操作：

将定期存款表内该笔存款的存款状态更新为 2，代表未到期取出。

在账户流水表中为定期账户插入一条取款记录，为活期账户插入一条代扣记录。

返回取款成功的提示信息。

如果存在已经到期的对应存款，我们选择最早的一笔进行取出，定位记录该笔存款的交易编号和存款利率：

记录定期账户和活期账户的旧余额。

在定期存款表内更新该笔存款状态为 1，表示到期取出。

在账户流水表中为定期账户插入一条取款记录，为活期账户插入一条利息结算记录。

返回取款成功的提示信息。

3）在信用卡账户下，需要判断取款后账户余额是否会小于-2000 元，不超过透支额度的情况下在账户流水表插入新的交易记录即可。

2.6 转账功能（transfer）

```
CREATE DEFINER=`root`@`%` PROCEDURE `transfer`(
    IN `cookie` BIGINT,
    IN `to_id` BIGINT,
    IN `stype` VARCHAR(2),
    IN `rtype` VARCHAR(2),
    IN `money` DECIMAL(20,2),
    OUT `state` INT,
    OUT `info` TEXT,
    OUT `balance` DECIMAL(20,2)
)
LANGUAGE SQL
NOT DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN
    DECLARE id BIGINT;
    DECLARE acc varchar(50);
    declare to_acc varchar(50);
    DECLARE rbalance DECIMAL(20,2);
    SELECT 卡号 INTO id FROM 临时会话 WHERE 会话号=cookie;
    if id IS NOT NULL then
    UPDATE 临时会话 SET 会话过期时间 =TIMESTAMPADD(minute,2,now()) WHERE 会话号=cookie;
        if money >10000 then
            set state = -1;
            SET info = '单次转账金额不得超过 10000 元';
        ELSE
            SELECT COUNT(*) INTO @ob FROM 银行卡密 WHERE 卡号=to_id;
            if @ob = 0 or @ob is null then
                set state = -2;
                SET info = '转账对方不存在';
            ELSE
```

```
                SELECT 账户号 into acc FROM 卡内账户 WHERE 卡号=id AND 账户类型=stype;
                SELECT 账户号 into to_acc FROM 卡内账户 WHERE 卡号=to_id AND 账户类型=rtype;
            SELECT 余额 INTO balance FROM 账户流水 WHERE 账户号 = acc  ORDER BY 交易时间 DESC LIMIT 1;
                if stype = '活期' or stype = '信用'then
                    if balance-(money*1.01) <0 and stype = '活期' then
                        set state = -2;
                        SET info = '余额不足以扣除转账金额与手续费，手续费率 1%';
                    elseif balance-(money*1.01) <-2000 and stype = '信用' then
                        set state = -2;
                        SET info = '透支额度不足以扣除转账金额与手续费，手续费率 1%';
                    else
                        INSERT INTO 账户流水(账户号,交易类型,映射账户,变动金额,余额,包含手续费)VALUES
                                    (acc,'转出',to_acc,-money*1.01,balance-money*1.01,money*0.01);
                    select max(交易编号) into @opid from 账户流水 where 账户号=acc and 交易类型='转出';
SELECT 余额 INTO @to_balance FROM 账户流水 WHERE 账户号 = to_acc  ORDER BY 交易时间 DESC LIMIT 1;
                        INSERT INTO 账户流水(账户号,交易类型,映射账户,映射交易号,变动金额,余额)VALUES
                                    (to_acc,'转入',acc,@opid,money,@to_balance+money);
            select max(交易编号) into @lastid from 账户流水 where 账户号=to_acc and 交易类型='转入';
                        UPDATE 账户流水 SET 映射交易号=@lastid WHERE 交易编号 = @opid;
                        if rtype = '定期' then
                            select 交易编号,交易时间 into @opid,@optime from 账户流水 where 账户号
=to_acc and 交易类型='转入' order by 交易时间 DESC LIMIT 1;
                            INSERT INTO 定期存款(交易编号,交易时间,账户号,存入金额,到期时间,日利
率)VALUES (@opid,@optime,to_acc,money,TIMESTAMPADD(day,30,@optime),0.0001);
                        end if;
                        set state= 1;
                        SET info = '转账成功';
                    END if;
                elseif stype ='定期' then
                    SELECT 账户号 into @huoqiacc FROM 卡内账户 WHERE 卡号=id AND 账户类型='活期';
SELECT 余额 INTO @huoqi_balance FROM 账户流水 WHERE 账户号 = @huoqiacc  ORDER BY 交易时间 DESC LIMIT 1;
                    if balance-money<0 then
                        set state = -2;
                        SET info = '余额不足以扣除转账金额';
                    elseif money*0.01 > @huoqi_balance then
                        set state = -2;
                        SET info = '活期账户余额（未结算定期利息）不足以扣除手续费，手续费率 1%';
                    else
SELECT COUNT(*) into @match_op from 定期存款 WHERE 存款状态=0 AND 存入金额=money AND 账户号=acc;
                        if @match_op is null or @match_op = 0 then
                            set state = -2;
                            SET info = '您的账户中没有与转账金额匹配的历史存款';
                        else
                            SELECT COUNT(*) into @match_op from 定期存款 WHERE 存款状态=0 AND 存入金额
=money and 到期时间<now() AND 账户号=acc;
                            if @match_op is null or @match_op = 0 then
                                select 交易编号 into @match_op from 定期存款 WHERE 存款状态=0 AND 存入
金额=money  AND 账户号=acc order by 到期时间 desc limit 1;
                                if @huoqi_balance <10 then
                                    set state = -2;
SET info = '活期账户余额不足！您的该笔定期存款尚未到期，转出存款需要从活期账户扣除 10 元手续费';
                                else
                                    UPDATE 定期存款 set 存款状态 = 2 where 交易编号=@match_op;
                                    insert into 账户流水(账户号,交易类型,映射账户,变动金额,余额,交易备
注)VALUES (acc,'转出',to_acc,0-money,balance-money,"转出未到期存款，于活期账户扣除手续费 10 元");
select max(交易编号) into @src_opid from 账户流水 where 账户号=acc and 交易类型='转出';
insert into 账户流水(账户号,交易类型,映射账户,映射交易号,变动金额,余额,包含手续费,交易备注)VALUES
(@huoqiacc,'代扣',acc,@src_opid,-10,@huoqi_balance-10,-10,"定期账户转出未到期存款，扣除手续费 10 元");
                                    select 余额 INTO @to_balance from 账户流水 where 账户号
=to_acc  ORDER BY 交易时间 DESC LIMIT 1;
                                    INSERT INTO 账户流水(账户号,交易类型,映射账户,映射交易号,变动金额,
余额)VALUES (to_acc,'转入',acc,@src_opid,money,@to_balance+money);
select max(交易编号) into @lastid from 账户流水 where 账户号=to_acc and 交易类型='转入';
                                    UPDATE 账户流水 SET 映射交易号=@lastid WHERE 交易编号 =@src_opid;
                                    if rtype = '定期' then
                                        select 交易编号,交易时间 into @opid,@optime from 账户流水 where
账户号=to_acc and 交易类型='转入' order by 交易时间 DESC LIMIT 1;
                                        INSERT INTO 定期存款(交易编号,交易时间,账户号,存入金额,到期时
间,日利率)VALUES (@opid,@optime,to_acc,money,TIMESTAMPADD(day,30,@optime),0.0001);
```

```
                                    end if;
                                    set state= 1;
                                    SET info = '转账成功';
                                end if;
                            else
SELECT 余额 INTO @dingqi_balance FROM 账户流水 WHERE 账户号 = acc ORDER BY 交易时间 DESC LIMIT 1;
SELECT 余额 INTO @huoqi_balance FROM 账户流水 WHERE 账户号 = @huoqiacc  ORDER BY 交易时间 DESC LIMIT 1;
                                select 交易编号,日利率 into @match_op,@lilv from 定期存款 WHERE 存款状
态=0 AND 存入金额=money order by 到期时间 asc limit 1;
                                UPDATE 定期存款 set 存款状态 = 1 where 交易编号=@match_op;
                                insert into 账户流水(账户号,交易类型,映射账户,变动金额,余额,交易备
注)VALUES (acc,'转出',to_acc,-money,@dingqi_balance-money,"转出到期存款，利息另转至活期账户");
select max(交易编号) into @src_opid from 账户流水 where 账户号=acc and 交易类型='转出';
                                insert into 账户流水(账户号,交易类型,映射账户,映射交易号,变动金额,余额,
包含手续费,交易备注)VALUES (@huoqiacc,'代扣',@huoqiacc,@src_opid,-money*0.01,@huoqi_balance-
money*0.01,-money*0.01,'定期账户转出到期存款，扣除手续费');
                                SELECT 余额 INTO @huoqi_balance FROM 账户流水 WHERE 账户号 =
@huoqiacc  ORDER BY 交易时间 DESC LIMIT 1;
                                insert into 账户流水(账户号,交易类型,映射账户,映射交易号,变动金额,余
额)VALUES (@huoqiacc,'定期利息',acc,@src_opid,@lilv*30*money, @huoqi_balance+@lilv*30*money);
SELECT 余额 INTO @to_balance FROM 账户流水 WHERE 账户号 = to_acc  ORDER BY 交易时间 DESC LIMIT 1;
                                INSERT INTO 账户流水(账户号,交易类型,映射账户,映射交易号,变动金额,余
额)VALUES (to_acc,'转入',acc,@src_opid,money,@to_balance+money);
select max(交易编号) into @lastid from 账户流水 where 账户号=to_acc and 交易类型='转入';
                                UPDATE 账户流水 SET 映射交易号=@lastid WHERE 交易编号 =@src_opid;
                                if rtype = '定期' then
                                        select 交易编号,交易时间 into @opid,@optime from 账户流水 where
账户号=to_acc and 交易类型='转入' order by 交易时间 DESC LIMIT 1;
                                        INSERT INTO 定期存款(交易编号,交易时间,账户号,存入金额,到期时
间,日利率)VALUES (@opid,@optime,to_acc,money,TIMESTAMPADD(day,30,@optime),0.0001);
                                end if;
                                set state= 1;
                                SET info = '转账成功';
                            end if;
                        end if;
                    END if;
                END if;
            END if;
        END if;
    END if;
SELECT 余额 INTO balance FROM 账户流水 WHERE 账户号 = acc  ORDER BY 交易时间 DESC LIMIT 1;
END
```

转账存储过程传入参数为会话号 cookie，对方卡号 id，己方账户类型 stype，对方账户类型 rtype，转账金额。传出参数为转账结果状态码 state，提示消息 info，余额 balance。

一次转账金额不能超过 10000 元，若超过，则需要返回失败提示。

在银行卡密表中查询转账对方是否存在，如果不存在，直接返回失败提示。

在卡内账户表中定位己方和对方的账户号 acc,to_acc 记录下来供后面使用。

1）如果转出账户类型为活期或者信用就比较容易：

将转账金额附加上 1%的手续费与余额进行比较，活期账户如果余额不足以扣取，信用账户如果如果扣完余额小于-2000，就返回余额不足的交易失败信息。

余额充足的情况下，为账户流水表插入一条己方账户的转出记录。

获取刚才插入的转出记录的交易编号，记录下来。

为对方账户插入一条转入记录，记录中的映射交易号填入刚刚记录的交易编号。

查询记录下刚才这条转入记录的交易编号，为己方的转出记录补充上映射交易号。

如果转入对方的账户类型是定期，那么还要额外在定期存款表为对方账户增加一条存款记录。

2）如果转出账户为定期账户：

在账户流水表检查定期账户的余额是否充足，如果不够就交易失败。

定期账户只能整存整取，所有的手续费和利息都由活期账户收支。故需检查该用户对应的活期账户是否足以扣除手续费，若不足则返回失败。

在定期存款表内查找是否有与转账金额相符的历史存款，如果没有就返回失败。

若有对应的存款，就查找有没有已经到期的存款。

2．1 如果存款都没有到期，就需要收取 10 元手续费，这里我们简化流程，如果已经扣除这笔未到期提款手续费了，就不再额外收取转账的 1%手续费。

检查活期账户余额是否足以扣除 10 元手续费，不足则返回失败。

活期余额充足的情况下正式开始转账操作：

更新定期存款表中该笔存款的存款状态为 2，表示未到期取出。

在账户流水表中为己方账户插入交易类型为转出的交易记录，该条记录不记录转出手续费。插入后获取并记录该条记录的交易编号@src_opid。

在账户流水表中为己方对应的活期账户插入一条交易类型为代扣的交易记录，填入包含手续费等信息，映射交易号为@src_opid。

在账户流水表为对方账户插入一条交易类型为转入的交易记录，该条交易记录的映射交易号也为@src_opid。插入后记录该条记录的交易编号，补充到转出记录的映射交易号中。

如果转入的账户是定期账户，还需要再定期存款表新增相应的存款信息。

自此可以返回转账成功消息。

2．2 如果有已经到期的存款，就可以直接开始转账操作，后面是核心的操作过程。

在定期存款表更新选定的该笔存款的存款状态为 1，表示到期取出。

在账户流水表为己方账户插入转出记录。

在账户流水表为己方活期账户插入代扣手续费记录。

在账户流水表为己方活期账户插入利息结算记录。

在账户流水表为对方账户插入转入记录。

若对方账户为定期账户，还要为该条转入信息额外在定期存款表插入新的存款记录。

2.7 查询余额功能（inquiry）

```
CREATE DEFINER=`root`@`%` PROCEDURE `inquiry`(
    IN `cookie` BIGINT,
    IN `ctype` VARCHAR(2),
    OUT `name` TEXT,
    OUT `balance` DECIMAL(20,2),
    OUT `flex_balance` DECIMAL(20,2)
)
LANGUAGE SQL
NOT DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN
```

```
    DECLARE id BIGINT;
    DECLARE day_money DECIMAL(20,2) DEFAULT 0;
    DECLARE acc VARCHAR(50);
    UPDATE 临时会话 SET 会话过期时间 =TIMESTAMPADD(minute,2,now()) WHERE 会话号=cookie;
    SELECT 卡号 INTO id FROM 临时会话 WHERE 会话号=cookie;
    if id IS NOT NULL then
        SELECT 身份证号 INTO @pid FROM 银行卡密 WHERE 卡号=id;
        SELECT 客户名 INTO name FROM 客户信息 WHERE 身份证号 = @pid;
        SELECT 账户号 INTO acc FROM 卡内账户 WHERE 卡号 = id AND 账户类型 = ctype;
        SELECT 余额 INTO balance FROM 账户流水 WHERE 账户号 =acc ORDER BY 交易时间 DESC
LIMIT 1;
        select SUM(变动金额) into day_money FROM 账户流水 WHERE TO_DAYS(交易时间) =
TO_DAYS(NOW()) AND 账户号=acc AND 交易类型='取款';
        if day_money IS NULL then
            set day_money = 0;
        END if;
        set flex_balance=5000+day_money ;
        if flex_balance > balance then
            SET flex_balance = balance;
        END IF;
    END if;
END
```

余额查询存储过程传入参数为会话号 cookie，账户类型 ctype，传出参数为客户名，账户余额和今日可取余额。

通过卡号 id 在账号卡密表定位到客户的身份证信息，以此在客户信息表查询到客户的名字存入 name。

在账户流水表中按照交易时间排序，取出最新的一条记录中的余额存入 balance。

使用聚集函数 sum 统计变动金额获得今日已经取出的金额，该值是负数，因此与 5000 相加即为今日剩余可取额度 flex_balance。但是若账户中的余额本身还不足 5000，就将余额作为取款额度返回。

2.8 查询交易记录功能（history）

```
CREATE DEFINER=`root`@`%` PROCEDURE `history`(
    IN `cookie` BIGINT,
    IN `ctype` VARCHAR(2),
    IN `cfrom` TIMESTAMP,
    IN `cto` TIMESTAMP,
    IN `corder` INT,
    IN `sc` int
)
LANGUAGE SQL
NOT DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN
    DECLARE id BIGINT;
    DECLARE acc VARCHAR(50);
    SELECT 卡号 INTO id FROM 临时会话 WHERE 会话号=cookie;
    if id IS NOT NULL then
        UPDATE 临时会话 SET 会话过期时间 =TIMESTAMPADD(minute,2,now()) WHERE 会话号=cookie;
        SELECT 账户号 into acc FROM 卡内账户 WHERE 卡号=id AND 账户类型=ctype;
        if corder > 0 then
            if sc =1 then
```

```
                SELECT 交易时间,交易类型,卡内账户.`卡号` 发起人或接收方,卡内账户.`账户类型` 发起人或接收方类型,变动金额 , 余额,交易备注 FROM 账户流水,卡内账户 WHERE 账户流水.`账户号` = acc and 卡内账户.`账户号`=账户流水.`映射账户` order BY 交易时间 DESC LIMIT corder;
            ELSE
                SELECT 交易时间,交易类型,卡内账户.`卡号` 发起人或接收方,卡内账户.`账户类型` 发起人或接收方类型,变动金额 , 余额 ,交易备注 FROM 账户流水,卡内账户 WHERE 账户流水.`账户号` = acc and 卡内账户.`账户号`=账户流水.`映射账户` order BY 交易时间 ASC LIMIT corder;
            END if;
        else
            if sc=1 then
                SELECT 交易时间,交易类型,卡内账户.`卡号` 发起人或接收方,卡内账户.`账户类型` 发起人或接收方类型,变动金额 , 余额,交易备注  FROM 账户流水,卡内账户 WHERE 账户流水.`账户号` = acc and 卡内账户.`账户号`=账户流水.`映射账户` and (交易时间 BETWEEN cfrom AND cto ) order BY 交易时间 DESC;
            ELSE
                SELECT 交易时间,交易类型,卡内账户.`卡号` 发起人或接收方,卡内账户.`账户类型` 发起人或接收方类型,变动金额 , 余额,交易备注  FROM 账户流水,卡内账户 WHERE 账户流水.`账户号` = acc and 卡内账户.`账户号`=账户流水.`映射账户` and (交易时间 BETWEEN cfrom AND cto ) order BY 交易时间 ASC;
            END if;
        END if;
    END if;
END
```

交易记录查询功能存储过程的传入参数为会话号 cookie，账户类型 ctype，查询起始日期 cfrom，查询截止日期 cto,查询条数 corder，排序方式 sc。无传出参数，直接返回结果集。

查询 corder 值为 0 时按照传入的查询起止时间生成结果集,返回所有记录。corder 的值大于 0 时返回按照查询结果最前的数条记录。

sc 指定查询结果的排序方式，为 1 时按照交易时间降序，即最新的记录排在最前面；为 0 时按照交易时间升序，最新的纪录排在最下面。

存储过程的并发控制说明：MySQL 的存储过程不是原子性的，这导致并发控制和异常中断可能出现问题。如果要优化该问题，可以选择更改数据库系统类型如换用 oracle，或者引入事务进行提交：

```
start transaction;
存储过程内容……
commit;
```

下方锁的控制不允许在存储过程中使用：

```
lock table tableName read;//读锁
lock table tableName write;//写锁
……
unlock tables;//所有锁表
```

2.9 定期清理临时会话事务

```
CREATE DEFINER=`root`@`%` EVENT `clean_session`
    ON SCHEDULE
        EVERY 1 MINUTE
    ON COMPLETION PRESERVE
```

```
    ENABLE
    COMMENT ''
    DO
DELETE FROM 临时会话 WHERE 会话过期时间 < NOW();
DELETE FROM 临时会话 WHERE 会话建立时间 < TIMESTAMPADD(MINUTE,-20,now());
```

除了创建存储过程外，数据库还创建了定时事务清理过期的会话，该定时器没 1 分钟执行一次。

删除临时会话表内会话过期时间已经早于当前时间的会话。

同时，这里强制设置一个会话持续建立不能超过 20 分钟。因此删除 20 分钟前建立的所有会话记录。

三、用户创建和权限授予

```
create user 'atm'@'%' identified by 'HNUdb';
flush privileges
GRANT EXECUTE ON PROCEDURE atm.signup TO atm@'%';
GRANT EXECUTE ON PROCEDURE atm.login TO atm@'%';
GRANT EXECUTE ON PROCEDURE atm.inquiry TO atm@'%';
GRANT EXECUTE ON PROCEDURE atm.history TO atm@'%';
GRANT EXECUTE ON PROCEDURE atm.modify TO atm@'%';
GRANT EXECUTE ON PROCEDURE atm.save TO atm@'%';
GRANT EXECUTE ON PROCEDURE atm.withdraw TO atm@'%';
GRANT EXECUTE ON PROCEDURE atm.transfer TO atm@'%';
```

前文所述的基本表、存储过程等项目结构均在 root 管理员用户下执行，这里新建一个名为 atm 的用户，许可登录 ip 不做限定，方便不同机器使用。

将前文创建的 8 个存储过程的执行权限授予用户 atm，这样一来，该用户经由执行 8 个存储过程的能力，不能自行查看、修改任一基本表内容。甚至该用户连查看存储过程的内容的权限也没有（查看存储过程需要授予存储过程的 select 权限），前端使用该账户操作数据库，安全性得到保障。