

课程编号：C0801207050

数据库应用程序设计实践 报告

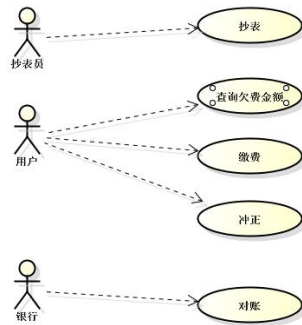


姓 名	XXXX	学 号	XXXX
班 级	XXXX	指 导 教 师	XXXX
开 设 学 期	2017-2018 第 二 学 期		
开 设 时 间	第 25 周 —— 第 26 周		
报 告 日 期	2018-08-30		
评 定 成 绩		评 定 人	
		评 定 日 期	

东北大学软件学院

1. 问题定义

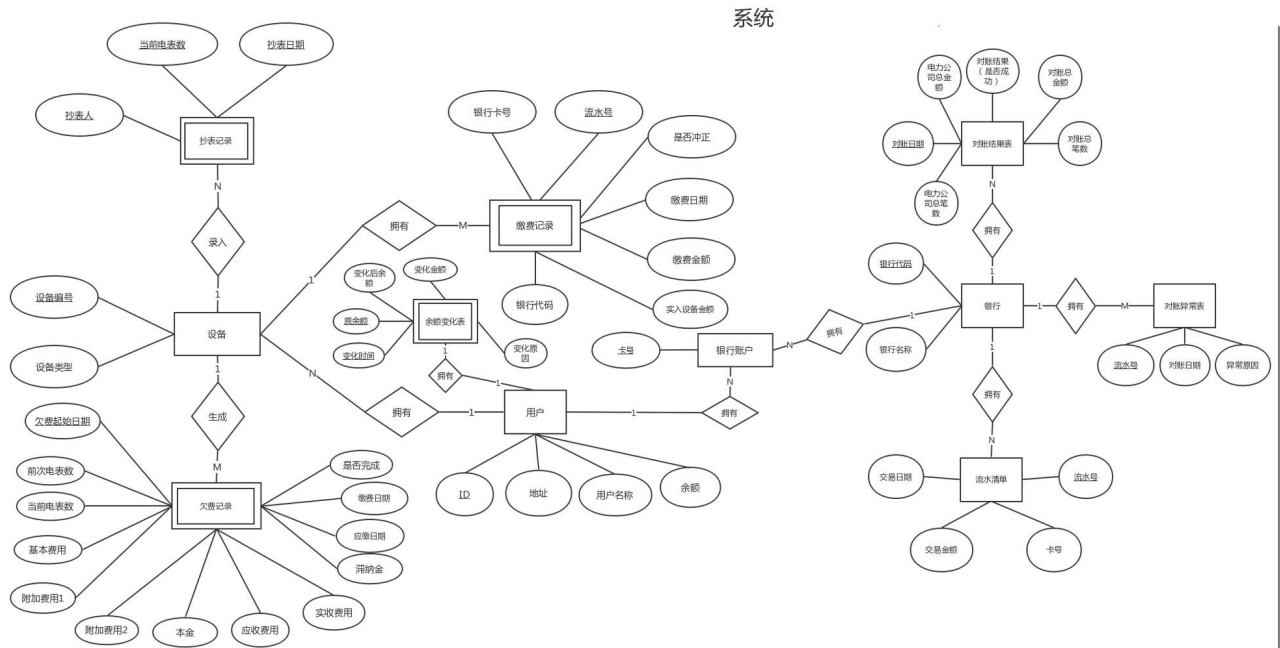
根据系统总体需求，该电力公司的系统需求大概分为五个总步骤



- 1) 抄表员抄表: 抄表员使用设备对各个用户进行抄表, 系统会主动进行计算, 记录用户的用电量以及产生的金额, 并自动生成账单, 供用户进行查询并进行缴费。每月抄表员进行抄表, 系统都能够自动进行扣费, 若该设备用户有余额, 则将金额划入账单。
- 2) 用户能够查询欠费金额: 用户能够查询自己的欠费, 以供进行缴费。可查询自己一台设备的欠费, 也能查询多台设备的欠费。
- 3) 用户缴费: 用户能够通过设备进行对应设备的缴费, 用户缴费时, 先使用自己原先的余额, 若余额充足, 则扣除余额的费用, 最后缴纳的金额再划入余额。若余额不充足, 则使用到缴纳金。此时需要实时更新余额变化表, 记录余额的变化。每个用户都有应缴费日期, 若超过应缴费日期, 则用户需要缴纳违约金; 若用户在缴费日期之前缴纳费用, 则只需支付本金。
- 4) 用户冲正: 若用户今日充错设备, 可在今日凌晨之前向公司发起冲正请求。将冲到别人设备中的金额拿回自己的前并冲到余额中去。
- 5) 通过银行发送的数据进行对账: 每日公司能够通过银行获取的流水清单, 进行对账。先对总账, 若总笔数以及总金额都相等, 说明今日的账目没有问题, 并记录进对账记录表。反之, 若哪个银行的账目无法对上, 则查询该银行的明细, 查询对账不齐的原因, 并记录进对账异常记录表格。

2. 数据库设计

请使用 E-R 图等方式描述你说设计的数据库及其表结构定义, 并给出充分的说明。



根据 ER 图，能够将关系转换为如下表格

1. 用户 (ID, 用户名, 用户余额, 地址)
2. 设备 (DID, 设备类型, ID)
3. 银行 (银行 ID, 银行名称)
4. 银行账户 (ID, 银行 ID, 卡号)
5. 余额变化记录 (ID, 变化时间, 原余额, 变化后余额, 变化金额, 变化原因)
6. 抄表记录 (抄表人, 当前电表数, 抄表日期, 设备编号)
7. 欠费记录 (欠费起始日期, 前次电表数, 当前电表数, 基本费用, 附加费用 1, 附加费用 2, 本金, 应收金额, 实收金额, 滞纳金, 应缴日期, 缴费日期, 是否完成, 设备编号)
8. 缴费记录 (流水号, DID, 银行 ID, 卡号, 支付金额, 实入设备金额, 支付日期, 是否冲正)
9. 对账结果记录 (对账日期, 公司笔数, 银行笔数, 公司对账金额, 银行对账金额, 是否成功, 银行 ID)
10. 对账异常记录 (流水号, 对账日期, 对账异常原因, 银行 ID)
11. 流水清单 (支付日期, 流水号, 支付金额, 银行卡号, 银行 ID)

根据分析，我认为，每台设备都应该具有自己的缴费记录，抄表记录以及欠费清单，而一个用户能够拥有台设备，以及多个银行账户，其中，每个用户都有余额变化记录来存储自己余额变化；而对于每个银行，系统内都应该有对应的表格来存储银行发来的记录。

3. 数据库端的系统实现

请描述你使用 SQL 和 PL/SQL 语句实现的系统功能，实现过程及其运行结果。

SQL 语句

- 1) 查询出所有欠费用户：选择欠费记录中，完成标志为否的部分

```
select distinct id, username, receivedAmount
from device join deviceuser using (ID) join arrearagerecord using (DID)
where finish=0;
```

```

133
134 -- 查询所有欠费用户
135 • select distinct id ,username, receivedAmount
136 from device join deviceuser using (ID) join arrearagerecord using(DID)
137 where finish=0;

```

id	username	receivedAmount
1000	裴行俭	0.00
1002	潘喜龙	0.00
1004	周叶夫	0.00

对于查询所有欠费用户，查询欠费清单，选出其中未完成的记录

2) 查询出拥有超过 2 个设备的用户

- a) select distinct id, username, count (*) as amount
from device join deviceuser using(id)
group by id
having amount>2;

```

65
66
67 • select distinct id, username, count(*) as amount
68 from device join deviceuser using(id)
69 group by id
70 having amount>2;
71 /*拥有大于两个设备的用户*/
72 /* 某人欠多少*/

```

id	username	amount
1002	潘喜龙	3

b) 顺便能够查询出欠费两个以上设备的用户

```

select distinct id, username
from (select *, count (*) as amount
      from device join deviceuser using(id)
      group by id
      having amount>2)
as T join arrearagerecord using (DID)
where finish=0
group by did;

```

```

56
57 -- 查询出欠费两个以上设备的用户
58 • select distinct id, username
59 from (select *, count(*) as amount
60       from device join deviceuser using(id)
61       group by id
62       having amount>2)
63 as T join arrearagerecord using(DID)
64 where finish=0
65 group by did;
66

```

id	username	amount
1002	潘喜龙	3

3) 统计电力企业某天的总已收费用，实收费用

```

4 set @selectedDay:='2018-08-30';
5 select @selectedDay as selectedDay, sum(capital), sum(overduefine),
6 sum(receivedAmount), sum(overduefine+capital) as actualFee, sum(receivedAmount), round(sum(capital+overduefine-receivedAmount),2) as 应收费用
7 from (select id, username, capital, (capital*0) as overduefine, receivedAmount, arrearageDate, payableDate, payDate, DID, finish
8 from device join deviceuser using (ID) join arrearagerecord using (DID)
9 where @selectedDay<payableDate and @selectedDay>=arrearageDate and finish=0
10 union
11 select id, username, capital, (capital*(datediff(@selectedDay, payableDate)*0.001) as overduefine,
12 receivedAmount, arrearageDate, payableDate, payDate, DID, finish
13 from device join deviceuser using (ID) join arrearagerecord using (DID)
14 where deviceType=01 and @selectedDay>payableDate and finish=0
15 union
16 select id, username, capital, (capital*(datediff(@selectedDay, date_sub(@selectedDay, interval dayofyear(now())-1 day))+1)*0.002
17 +capital*(datediff(date_sub(@selectedDay, interval dayofyear(@selectedDay)-1 day), payableDate))*0.003) as overduefine,
18 receivedAmount, arrearageDate, payableDate, payDate, DID, finish
19 from device join deviceuser using (ID) join arrearagerecord using (DID)
20 where deviceType=02 and year(payableDate)<year(@selectedDay) and finish=0
21 union
22 select id, username, capital, (capital*(datediff(@selectedDay, date_sub(@selectedDay, interval dayofyear(@selectedDay)-1 day))+1)*0.002) as overduefine,
23 receivedAmount, arrearageDate, payableDate, payDate, DID, finish
24 from device join deviceuser using (ID) join arrearagerecord using (DID)
25 where deviceType=02 and year(payableDate)=year(@selectedDay) and finish=0
26 union
27 select id, username, capital, overduefine, receivedAmount, arrearageDate, payableDate, payDate, DID, finish
28 from device join deviceuser using (ID) join arrearagerecord using (DID)
29 where finish=1) T
30 where finish=0
31

```

selectedDay	sum(capital)	sum(overduefine)	sum(receivedAmount)	actualFee	sum(receivedAmount)	应收费用
2018-08-30	209.89	12.87388	0.00	222.76388	0.00	222.76

4) 查询出所有欠费超过半年的用户

```

39
40 -- 查询出欠费半年的用户
41 select distinct id, username, did, arrearageDate
42 from device join deviceuser using (ID) join arrearagerecord A using (DID)
43 where beginMeterAmount=0 and finish=0 and datediff(@selectedDay, arrearageDate)>182;
44
45

```

id	username	did	arrearageDate
1000	韩行伦	2000	2018-01-05
1002	潘晋龙	2003	2018-06-05
1004	周叶夫	2007	2018-07-01

5) 查询任意用户的欠费总额

```

73 set @id=1002;
74 select @selectedDay as 指定日期, username, sum(capital), sum(overduefine), sum(receivedAmount), payableDate, sum(overduefine+capital) as actualFee,
75 DID, sum(round(capital+overduefine-receivedAmount,2)) as 应收费用
76 from (select id, username, capital, (capital*0) as overduefine, receivedAmount, arrearageDate, payableDate, payDate, DID, finish
77 from device join deviceuser using (ID) join arrearagerecord using (DID)
78 where @selectedDay<payableDate and @selectedDay>=arrearageDate and finish=0
79 union
80 /*01设备*/
81 /*超过应缴日期未还款*/
82 select id, username, capital, (capital*(datediff(@selectedDay, payableDate)*0.001) as overduefine,
83 receivedAmount, arrearageDate, payableDate, payDate, DID, finish
84 from device join deviceuser using (ID) join arrearagerecord using (DID)
85 where deviceType=01 and @selectedDay>payableDate and finish=0
86 union
87 /*02设备*/
88 /*跨年*/
89 select id, username, capital, (capital*(datediff(@selectedDay, date_sub(@selectedDay, interval dayofyear(now())-1 day))+1)*0.002
90 +capital*(datediff(date_sub(@selectedDay, interval dayofyear(@selectedDay)-1 day), payableDate))*0.003) as overduefine,
91 receivedAmount, arrearageDate, payableDate, payDate, DID, finish
92 from device join deviceuser using (ID) join arrearagerecord using (DID)
93 where deviceType=02 and year(payableDate)<year(@selectedDay) and finish=0
94 union
95 /*未跨年*/
96 select id, username, capital, (capital*(datediff(@selectedDay, date_sub(@selectedDay, interval dayofyear(@selectedDay)-1 day))+1)*0.002) as overduefine,
97 receivedAmount, arrearageDate, payableDate, payDate, DID, finish
98 from device join deviceuser using (ID) join arrearagerecord using (DID)
99 where deviceType=02 and year(payableDate)=year(@selectedDay) and finish=0
100 union
101 select id, username, capital, overduefine, receivedAmount, arrearageDate, payableDate, payDate, DID, finish
102 from device join deviceuser using (ID) join arrearagerecord using (DID)
103 where finish=1) T
104 where finish=0 and T.id=@id;
105

```

指定日期	username	sum(capital)	sum(overduefine)	sum(receivedAmount)	payableDate	actualFee	DID	应收费用
2019-02-20	潘晋龙	101.19	21.98723	0.00	2018-07-31	123.17723	2003	123.18

6) 查询出某个月用电量最高的3名用户


```

47
48 -- 查询出某月用电量最高的用户
49 • set @theMonth:=7;
50 • select id, username, (currentMeterAmount-beginMeterAmount) as 用电量
51 from device join deviceuser using (ID) join arrearagerecord A using(DID)
52 where month(paymentDate)=@theMonth
53 group by id
54 order by currentMeterAmount-beginMeterAmount desc
55 LIMIT 3;

```

id	username	用电量
1002	潘喜龙	100
1004	周平夫	98
1003	叶香	64

7) 查询出电力企业某个月哪天的缴费人数最多

```

106 /*某个月缴费人数最多的一天*/
107 • select month(paymentDate), paymentDate, count(paymentDate) as amount
108 from paymentrecord
109 group by paymentDate
110 having (amount)>=all(select count(paymentDate) as amount
111 from paymentrecord
112 group by paymentDate
113 );

```

month(paymentDate)	paymentDate	amount
8	2018-08-20	2

8) 按设备类型使用人数从高到低排序查询出设备类型，使用人数

```

116
117 /*按设备类型使用人数从高到低排序查询列出设备类型，使用人数。*/
118 • select deviceType, count(distinct id) as amountOfUsers
119 from device
120 group by deviceType;

```

deviceType	amountOfUsers
01	4
02	3

9) 统计某个月各银行缴费人次，从高到低排序

```

120
121 group by deviceType;
122 /*统计每个月各银行缴费人次，从高到低排序。*/
123 • set @month:=8;
124 • select @month, bankID, count(bankID) as amount from serialrecord where month(paymentDate)=@month group by bankID order by count(bankID) desc;

```

@month	bankID	amount
8	CMB	2
8	ICBC	1

10) 查询出电力企业所有新增用户（使用设备不足半年）

```

126
127 -- 所有新增用户
128 • set @selectedDay='2018-09-02';
129 • select distinct id, username, did, arrearageDate
130 from device join deviceuser using (ID) join arrearagerecord A using(DID)
131 where beginMeterAmount=0 and datediff(@selectedDay,arrearageDate)<182;

```

id	username	did	arrearageDate
1001	秦冉平	2002	2018-03-05
1002	潘喜龙	2003	2018-06-05
1004	周平夫	2007	2018-07-01
1003	叶香	2006	2018-07-07

存储过程

1) 查询

查询前，先进行是否有此用户的判断

```
if (recordNumber=0) then
    set notice='不存在此客户';
    select notice;
else -- 找到此用户
```

如果设备欠费，进行分类判断

a) 如果设备在应缴费日期之前，则无滞纳金

```
select id,address, username, capital, (capital*0) as overduefine, receivedAmount, arrearageDate, payableDate, payDate, DID, finish
from device join deviceuser using (ID) join arrearagerecord using (DID)
where selectedDay<=payableDate and selectedDay>=arrearageDate and finish=0
union
```

b) 对于设备 01，如果超过应缴费日期

```
/*超过应缴日期未还款*/
select id,address, username, capital, (capital*datediff(selectedDay,payableDate)*0.001) as overduefine,
receivedAmount, arrearageDate, payableDate, payDate, DID, finish
from device join deviceuser using (ID) join arrearagerecord using (DID)
where deviceType=01 and selectedDay>payableDate and finish=0
union
/*未还款*/
```

c) 对于设备 02，如果超过应缴费日期

```
/*02设备*/
/*跨年*/
select id,address, username, capital, (capital*(datediff(selectedDay,date_sub(selectedDay,interval dayofyear(now())-1 day))+1)*0.002
+capital*(datediff(date_sub(selectedDay,interval dayofyear(selectedDay)-1 day),payableDate))*0.003) as overduefine,
receivedAmount, arrearageDate, payableDate, payDate, DID, finish
from device join deviceuser using (ID) join arrearagerecord using (DID)
where deviceType=02 and year(payableDate)<year(selectedDay) and finish=0
union
/*未跨年*/
select id,address, username, capital, (capital*(datediff(selectedDay,date_sub(selectedDay,interval dayofyear(selectedDay)-1 day))+1)*0.002) as overduefine,
receivedAmount, arrearageDate, payableDate, payDate, DID, finish
from device join deviceuser using (ID) join arrearagerecord using (DID)
where deviceType=02 and year(payableDate)=year(selectedDay) and finish=0
union
```

2) 缴费

a) 首先进行输入判断

```
if(countDID=0) then
    set notice='输入错误，本公司无此设备';
    select notice;
else -- 输入正确，选择卡号进行缴费
    select count(*) into countAccount from bankaccount where accountNumber=p_accou
    if(countAccount=0) then
        set notice='输入错误';
        select notice;
    else -- 进行缴费
        if(p_paymentAmount<0) then
            set notice='缴费金额错误，金额不能为负数';
            select notice;
```

b) 首先进行判断是否是给自己缴费，如果是，则动用到自己的余额；否则，不使用到设备对应用户的余额部分

```
if(p_ID=p_ID_to) then
    select balance into userBalance from deviceuser where ID=p_ID; -- 获得我的用户余额
else
    set userBalance=0;
end if;
```

c) 开始判断该用户有多少笔欠费，循环，对于欠费账单，从最远的那笔账单开始缴费。

若付款和余额（若交给别人则为 0）相加足够支付这笔账单：其中，若余额不够支付，则意味着所有余额都进入账单，并且需要动用到支付金额
反之，只需要扣除余额

更新余额变化记录表（若原余额为 0，则无需更新），更新欠费账单

操作余额以及用户支付款

```

if((p_paymentAmount+userBalance)>=(actualFee-p_receivedAmount)) then -- 如果付的费用能够支付这个月的
if (userBalance<(actualFee-p_receivedAmount)) then -- 余额不足支付，则用到支付金
-- 更新余额变化表
jump_label1: begin
if(userBalance=0) then
leave jump_label1;
end if;
insert into balancechangesrecord(ID,changeTime,beforeChange,afterChange,changeAmount,reason)
values(p_ID, current_timestamp(),userBalance,0, userBalance,'01');

end jump_label1;
set p_paymentAmount=p_paymentAmount-((actualFee-p_receivedAmount)-userBalance);
set userBalance=0;
else
insert into balancechangesrecord(ID,changeTime,beforeChange,afterChange,changeAmount,reason)
values(p_ID, current_timestamp(),userBalance,userBalance-(actualFee-p_receivedAmount),(actualFee-p_receivedAmount),'01');
set userBalance=userBalance-(actualFee-p_receivedAmount);
end if;

update arrearagerecord
set receivedAmount=actualFee,actualReceivebleAmount=actualFee,payDate=selectedDay,overdueFine=(actualFee-capital),finish=1
where DID=p_DID and payableDate=p_payableDate;

```

- d) 若余额和付款都不够支付这笔账单，则，所有钱都进入到账单中，更新余额变化记录表（原余额为 0 不用更新），更新欠费清单的已交金额。当付款值最后操作到 0，则退出循环

```

else if((p_paymentAmount+userBalance)<(actualFee-p_receivedAmount)) then -- 费用不够支付这个月的
jump_label2: begin
if(userBalance=0) then
leave jump_label2;
end if;
insert into balancechangesrecord(ID,changeTime,beforeChange,afterChange,changeAmount,reason)
values(p_ID, current_timestamp(),userBalance,0, userBalance,'01');
end jump_label2;

update arrearagerecord
set receivedAmount=p_receivedAmount+(p_paymentAmount+userBalance)
where DID=p_DID and payableDate=p_payableDate;

set p_paymentAmount=0;
set userBalance=0;
end if;
if(p_paymentAmount<=0) then -- 如果支付金额已经未0，则直接跳出循环
leave outer_label;
end if;

```

- e) 更新缴费记录表，更新用户余额，余额变化表（若交给别人最后有剩余）

```

insert into paymentrecord (serialNumber,DID,bankID,accountNumber,paymentAmount,inDeviceAmount,paymentDate,reversal)
values(serialNumber,p_DID,p_bankID,p_accountNumber,p_paymentAmount2,(userBalance2-userBalance)+(p_paymentAmount2-p_paymentAmount),selectedDay,false);
if((userBalance+p_paymentAmount)>0) then -- 如果余额>0 即，未使用银行转入的金额，多余的金额加入账户余额，更新用户余额，插入余额变化记录
if(p_ID>p_ID_to) then
select balance into userBalance from deviceuser where ID=p_ID;
end if;
insert into balancechangesrecord(ID,changeTime,beforeChange,afterChange,changeAmount,reason)
values(p_ID, current_timestamp(),userBalance,userBalance+p_paymentAmount, p_paymentAmount,'02');
end if;

update deviceuser set balance=userBalance+p_paymentAmount
where ID=p_ID;

```

3) 冲正

- a) 首先，从银行找到自己要冲正的流水号，并判断公司是否有此流水号。
b) 对于付款者，只需要找到自己为别人交了多少钱，将钱拿回。
更新自己的余额，更新自己的余额变化表（若并未给别人交上款，则无需更新）
公司更新此人的缴费记录，将交错的那笔账记为冲正。

```

-- select count(*) from paymentrecord where paymentDate=selectedDay; -- 找到今天所有的支付记录条数
-- 对于付款者 可找到自己付款的卡号，付款的到别人账上的总金额
select accountNumber,paymentAmount,round(inDeviceAmount,2),bankID into p_accountNumber_to,p_paymentAmount_to ,p_inDeviceAmount_to,p_bankID
from paymentrecord
where serialNumber=p_serialNumber;
-- 只需要拿回p_inDeviceAmount_to 便可
select balance into userBalance_to from deviceuser where ID=p_ID_to; -- 此人原本有多少余额
update deviceuser
set balance=userBalance_to+p_inDeviceAmount_to
where ID=p_ID_to;

if(p_inDeviceAmount_to>0) then
insert into balancechangesrecord(ID, changeTime,beforeChange,afterChange,changeAmount,reason)
values (p_ID_to,current_timestamp(),userBalance_to,userBalance_to+p_inDeviceAmount_to,p_inDeviceAmount_to,'02');
end if;
update paymentrecord
set reversal=1
where serialNumber=p_serialNumber;

```


- c) 对于收款者, 退出今日所有交到自己该设备的金额回余额, 扣除付款者交的金额则为今日自己原本的余额以及缴纳的费用。
循环, 逐笔将欠费金额恢复原样。更新欠费记录表。

-- 更新欠费记录表回原样

```
if(p_inDeviceAmount_thisTotal<=p_receivedAmount) then
    set p_receivedAmount=p_receivedAmount-p_inDeviceAmount_thisTotal;
    set p_inDeviceAmount_thisTotal=0;
else
    set p_inDeviceAmount_thisTotal=p_inDeviceAmount_thisTotal-p_receivedAmount;
    set p_receivedAmount=0;
end if;

update arrearagerecord
set actualReceivebleAmount=null, receivedAmount=p_receivedAmount, overdueFine=null, payDate=null, finish=0
where DID=p_DID_this and receivedAmount<>0 and payableDate=p_payableDate
order by payableDate desc;
if( p_inDeviceAmount_thisTotal<=0) then
    leave out_label;
end if;
```

- d) 将退回的那部分钱回缴, 类似于抄表时的余额缴费 (下附于抄表过程)

4) 对总账

- a) 判断银行总笔数和公司总笔数是否相同, 银行总转账金额和公司总收取金额是否相同。插入今日的对账记录

```
insert into resultrecord(checkDate, companyCheckingTotal, checkingTotal, companyCheckingAmount, checkingAmount, success, bankID)
values(p_checkDate, p_companyCheckingTotal, p_checkingTotal, p_companyCheckingAmount, p_checkingAmount, true, p_bankID);-- 如果成功, 则插入成功的记录
select curdate();
if(! (p_companyCheckingTotal=p_checkingTotal&&p_companyCheckingAmount=p_checkingAmount)) then
    update resultrecord
    set success=false
    where checkDate=p_checkDate and bankID=p_bankID;
```

5) 对细账

- a) 若对某银行总账结果于本公司不相符, 则进行细账查对首先比较银行笔数和公司笔数谁多, 使用多的一方进行查对。
b) 使用游标循环判断

```
DECLARE myCursor1 cursor for
select serialNumber, paymentAmount, accountNumber, bankID
from serialrecord
where paymentDate=DATE_SUB(p_checkDate, INTERVAL 1 DAY);
DECLARE myCursor2 cursor for
select serialNumber, paymentAmount, accountNumber, bankID
from paymentrecord
where paymentDate=DATE_SUB(p_checkDate, INTERVAL 1 DAY);
DECLARE continue HANDLER FOR NOT FOUND SET done=1;
```

- c) 首先判断对应方是否有相应的流水号, 若有, 进行判断金额是否相对应。找到不对应一方, 记录原因,

```
insert into resultrecord(checkDate, companyCheckingTotal, checkingTotal, companyCheckingAmount, checkingAmount, success, bankID)
values(p_checkDate, p_companyCheckingTotal, p_checkingTotal, p_companyCheckingAmount, p_checkingAmount, true, p_bankID);-- 如果成功, 则插入成功的记录
select curdate();
if(! (p_companyCheckingTotal=p_checkingTotal&&p_companyCheckingAmount=p_checkingAmount)) then
    update resultrecord
    set success=false
    where checkDate=p_checkDate and bankID=p_bankID;
```

6) 抄表

- a) 首先进行判断, 抄表金额为现金额, 必须大于原金额

```
if(p_currentMeterAmount<p_beginMeterAmount) then
    set notice='输入错误';
    select notice;
    leave exit_label;
end if;
```

- b) 生成抄表记录, 并计算各种费用, 然后生成新的账单记录

```
-- 生成抄表记录
insert into meterrecord(DID, dateOfReadMeter, currentMeterAmount, meterReader)
values(p_DID, p_dateOfReadMeter, p_currentMeterAmount, p_meterReader);
```

```

set p_basicAmount=(p_currentMeterAmount-p_beginMeterAmount)*0.46;
insert into arrearagerecord(DID,arrearageDate,beginMeterAmount,currentMeterAmount,basicAmount,appendAmount1,
                           appendAmount2,capital,receivedAmount,payableDate)
values(p_DID,p_dateOfReadMeter,p_beginMeterAmount,p_currentMeterAmount,p_basicAmount,p_basicAmount*0.08,
       p_basicAmount*appendFee2Rate,p_basicAmount+ (p_basicAmount*0.08) +p_basicAmount*appendFee2Rate,0,
       date_sub(date_add(p_dateOfReadMeter - day(p_dateOfReadMeter) +1,interval 1 month ),interval 1 day));

```

- c) 若该设备用户有余额，则开始进行余额扣费，直到余额扣光或账单支付完毕。类似正常缴费，从最远的账单开始缴费，若足够支付这笔账单，则继续进行余额缴费，此账单完成，更新账单，以及余额变化记录，继续寻找下一笔账单；反之，则余额扣光，更新账单已收费用，缴费过程停止，更新余额变化记录。

```

limit 0,1;
-- 开始进行缴费操作
set shouldPay=p_capital+p_overduefine-p_receivedAmount;
if(userBalance>shouldPay) then -- 如果余额足够支付
    set userBalance=userBalance-shouldPay;
    update arrearagerecord
    set receivedAmount=p_capital+p_overduefine,actualReceivebleAmount=p_capital+p_overduefine,payDate=p_dateOfReadMeter,overdueFine=p_overduefine,finish=1
    where DID=p_DID and payableDate=p_payableDate;
else
    update arrearagerecord
    set receivedAmount=userBalance
    where DID=p_DID and payableDate=p_payableDate;
    set userBalance=0;
end if;
insert into balancechangesrecord(ID, changeTime,beforeChange,afterChange,changeAmount,reason)
values(p_ID, current_timestamp(),userBalanceBefore,userBalance,userBalanceBefore-userBalance,'01');
set userBalanceBefore=userBalance;
set i=i+1;
end while;
end out_label;
end if;

```

- d) 最后更新用户的余额

4. 程序实现

请描述你使用了那些高级语言，如Java语言，与数据库进行连接，并实现了系统的哪些功能，以及实现过程和结果。

使用了JDBC，使用java体现。

- 1) 首先进行注册驱动，调用驱动器

```

// 1. 注册驱动
ResultSet res = null;
try {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
    } catch (ClassNotFoundException e) {
        System.out.println("Driver could not be loaded");
        System.exit( status: 0);
    }
}

```

- 2) 获取数据库连接

```

// 1. 注册驱动
ResultSet res = null;
try {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
    } catch (ClassNotFoundException e) {
        System.out.println("Driver could not be loaded");
        System.exit( status: 0);
    }
}

```

3) 调用存储过程抄表

```

/*存储过程部分*/
public static void readMeter(Connection conn,String dateOfReadMeter,int DID,int currentMeterAmount,String meterReader) throws SQLException {
    java.sql.Date date=strToDate(dateOfReadMeter);
    //存储过程函数固定格式: {call xxx}
    CallableStatement cs = conn.prepareCall( sql: "{call meterRead(?, ?, ?, ?)}");
    cs.setDate( parameterIndex: 1, date);
    cs.setInt( parameterIndex: 2, DID);
    cs.setInt( parameterIndex: 3, currentMeterAmount);
    cs.setString( parameterIndex: 4, meterReader);
    cs.executeUpdate();
}

```

4) 调用存储过程支付

```

247 public static void pay(Connection conn, int ID, int accountNumber, double paymentAmount, int DID, String day) throws SQLException {
248     java.sql.Date date=strToDate(day);
249     CallableStatement cs = conn.prepareCall( sql: "{call pay(?, ?, ?, ?, ?)}");
250     cs.setInt( parameterIndex: 1, ID);
251     cs.setInt( parameterIndex: 2, accountNumber);
252     cs.setDouble( parameterIndex: 3, paymentAmount);
253     cs.setInt( parameterIndex: 4, DID);
254     cs.setDate( parameterIndex: 5, date);
255     cs.executeUpdate();
256     System.out.println(date);
257 }
258

```

5) 调用存储过程查询

```

58
59 public static void search(Connection conn, int ID, String day) throws SQLException {
60     java.sql.Date date=strToDate(day);
61     System.out.println(date);
62     CallableStatement cs = conn.prepareCall( sql: "{call search(?, ?)}");
63     cs.setInt( parameterIndex: 1, ID);
64     cs.setDate( parameterIndex: 2, date);
65     cs.executeUpdate();
66     ResultSet res=cs.executeQuery();
67     System.out.println("用户名 地址 应收费用 实收费用");
68     while (res.next()) {
69         String username=res.getString( columnLabel: "username");
70         String address=res.getString( columnLabel: "address");
71         double shouldPay =res.getDouble( columnLabel: "shouldPay");
72         double alreadyPay=res.getDouble( columnLabel: "alreadyPay");
73         System.out.println(username + " " +address+ " "+shouldPay+ " "+alreadyPay);
74     }
75     System.out.println(date);
76 }
77
Main > pay()
un Main

```

6) 调用存储过程对总账（若有问题自动对细账）

```

public static void check(Connection conn,String bankID,int checkingTotal,double checkingAmount,String day) throws SQLException {
    java.sql.Date date=strToDate(day);
    CallableStatement cs = conn.prepareCall( sql: "{call checkTotal(?, ?, ?, ?)}");
    cs.setString( parameterIndex: 1, bankID);
    cs.setInt( parameterIndex: 2, checkingTotal);
    cs.setDouble( parameterIndex: 3, checkingAmount);
    cs.setDate( parameterIndex: 4, date);
    cs.executeUpdate();
    System.out.println(date);
}

```

7) 调用存储过程冲正

```

public static void reverse(Connection conn, int serialNumber, String day) throws SQLException {
    java.sql.Date date = strToDate(day);
    CallableStatement cs = conn.prepareCall( sql: "{call reverse(?,?)}");

    cs.setInt( parameterIndex: 1, serialNumber);
    cs.setDate( parameterIndex: 2, date);
    cs.executeUpdate();
    System.out.println(date);
}

```

执行 SQL 语句，连接 JDBC

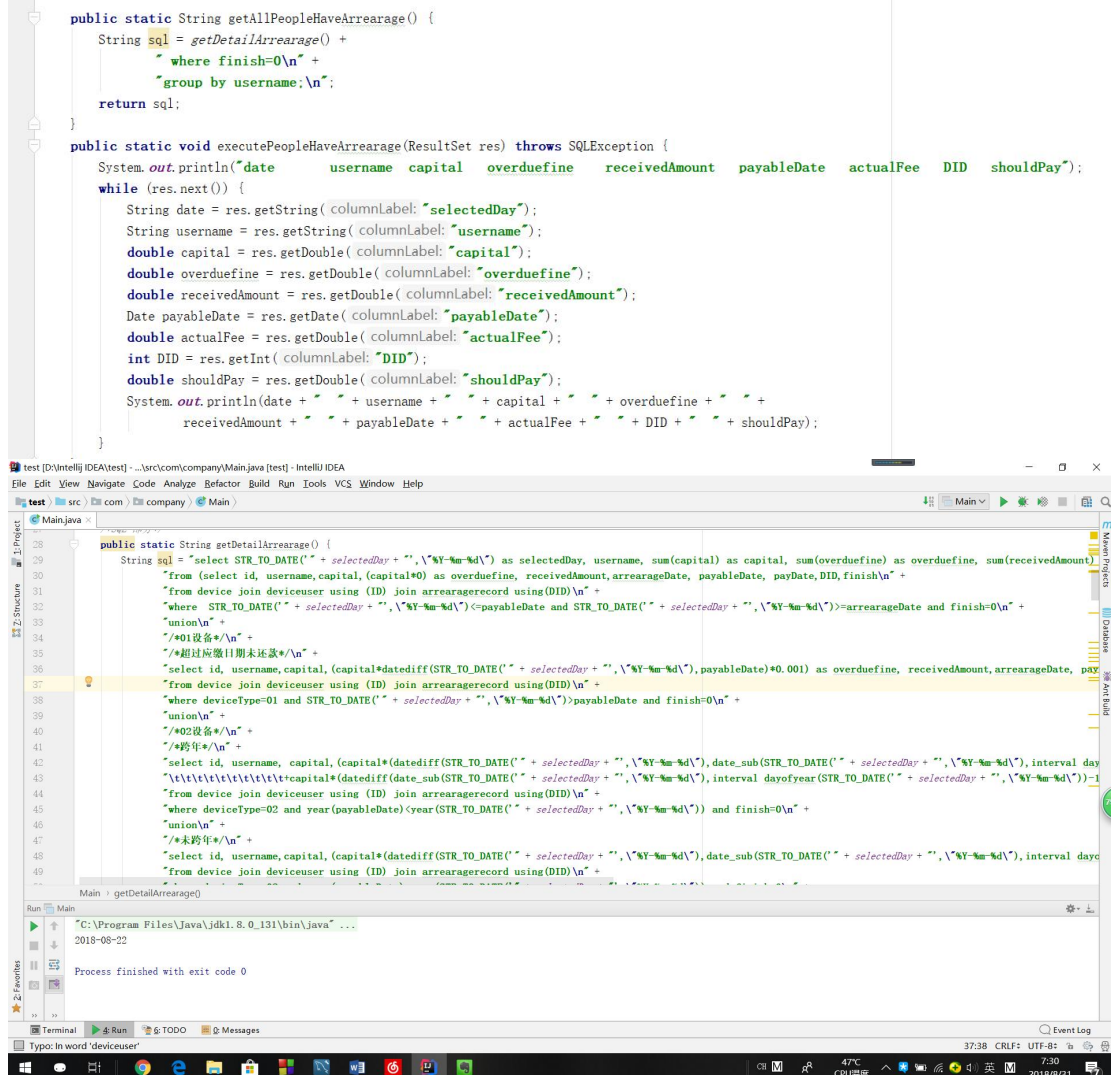
8) 获得所有欠费的用户

```

public static String getAllPeopleHaveArrearage() {
    String sql = getDetailArrearage() +
        "where finish=0\n" +
        "group by username;\n";
    return sql;
}

public static void executePeopleHaveArrearage(ResultSet res) throws SQLException {
    System.out.println("date\tusername\tcapital\toverduefine\treceivedAmount\tpayableDate\tactualFee\tDID\tshouldPay");
    while (res.next()) {
        String date = res.getString( columnLabel: "selectedDay");
        String username = res.getString( columnLabel: "username");
        double capital = res.getDouble( columnLabel: "capital");
        double overduefine = res.getDouble( columnLabel: "overduefine");
        double receivedAmount = res.getDouble( columnLabel: "receivedAmount");
        Date payableDate = res.getDate( columnLabel: "payableDate");
        double actualFee = res.getDouble( columnLabel: "actualFee");
        int DID = res.getInt( columnLabel: "DID");
        double shouldPay = res.getDouble( columnLabel: "shouldPay");
        System.out.println(date + "\t" + username + "\t" + capital + "\t" + overduefine + "\t" +
            receivedAmount + "\t" + payableDate + "\t" + actualFee + "\t" + DID + "\t" + shouldPay);
    }
}

```



其中，sql 语句与上述相同，只是将其改变为 java 代码，此处以此为演示。

5. 遇到的问题及其解决方案

请描述你在整个实践过程中遇到的主要问题，及其解决方案和效果。

- 1) 冲正时：原表难以知道到底为他人冲了多少钱
 - a) 产生原因：原本只有余额记录表能够查看到更新变化，然而由于可能此人可能

缴纳了几笔费用才发现自己缴费错误，难以定位到自己缴费错误时的余额变化。

- b) 解决方案：我为我的缴费记录表中添加了一个属性为 `inDeviceAmount`，在每次进行缴费的时候都能够进行计算。不管是否为自己缴费，在插入缴费记录时都计算最终进入设备的金额有多少，以供之后冲正的方便。
- 2) 为查询方便使用了 `view`：发现 `view` 不会根据我的操作进行实时改变
- a) 产生原因：经查询，了解到当我的视图中包含聚合函数，`distinct`，`group by`，`union`，`having` 以及各种子查询时，视图是不会实时跟着更新的。
 - b) 解决方案，尽量少使用视图，视图可能会导致性能下降。能够将值存下来，尽量少每次都使用视图
- 3) 使用游标时：游标使用时最后一行数据读取了两次
- a) 产生原因：经查询，游标只有在执行到 `fetch` 指令的时候才会进行判断是否结束，原先将 `fetch` 语句放在 `repeat` 下行，导致游标读到最后一行时必须再循环一次才能做到判断。
 - b) 解决方案：在开始循环前先进行一次 `fetch` 操作，然后执行我的 `sql` 语句，在进行下一次 `repeat` 再进行 `fetch` 操作，这样便能够在每次循环开始前进行判断是否需要停止循环了
- 4) 使用游标时：游标循环次数和理想次数不同。
- a) 产生原因：询问老师后，了解到，游标每次的声明是静态的，意味着，每次 `declare` 后，游标的循环次数就已经固定了，之后我对表进行增删操作都回影响到游标的使用。
 - b) 解决方案：游标不是万能的，要根据情况而定。

6. 创新点

请描述在你完成的实践任务中，有哪些创新之处。

- 1) 为缴费记录增加了一个 `inDeviceAmount` 的属性，方便冲正的时候进行计算，这样我就无需查询余额变化记录来了解变化，也方便计算每次缴费操作之后余额的变化，节省了很大的工作量。
- 2) 为缴费记录增添了一个标志 `reversal`，而不是使用文件提供的交负值的方法。根据原则，缴费不能有负值，使用标志方便了之后的对账以及判断。
- 3) 将存储过程分开，让一个存储过程调用另一个存储过程，提高了效率。比如，当在对账的时候，当对账出错，便自动调用对明细的存储过程，不用自己手动进行对明细，提高了效率。

7. 总结

请对你所完成的实践成果做一个总结。

- 1) 通过这次实验，充分学习和了数据库的相关知识，从需求业务分析来画出 ER 图，设计表结构
- 2) 练习了 `sql` 语句以及事物存储过程，大大锻炼了自己，学习了很多新的内容。
- 3) 测试是很重要的！每次执行完一个操作我都会进行一次测试，如此减少了因为前序的小错误导致的后面程序的大面积瘫痪（前车之鉴）。
- 4) 这次的作业让我提升了很多，很感谢老师，通过和老师同学的交流，了解他人的想法，有助于我换位思考。比如，一开始想冲正的过程的时候考虑过于复杂，通过和同学的交流，清晰了思路。

参考文献

例:

[1] 萨师煊, 王珊.数据库系统概论 (第三版) [M].北京: 高等教育出版社,2002,122-150.

[2] 侯捷.Java 编程思想[M].北京:机械工业出版社,2002,22-35.

评价标准和评分依据:

评价内容	具体要求	评分依据	评分标准
分析问题能力	按实践课要求, 认真分析需求规约, 界定好系统需求范围, 设计系统的E-R图, E-R图设计合理, 能将概念模型转换给关系模式, 满足系统需求。	1. 表中字段的设计至少要符合数据库理论中三个范式的要求。 2. 表的数量足够业务需求, 太多和太少都不行。 3. 表的设计能够覆盖 任务书9.4节10个题目所需的查询。 4. 表之间的关系描述正确。 5. 有设计数据字典, 存放基础数据(基础数据从字典表获取, 基础数据改变时不需要改动plsql代码)。 6. 设计中考虑到了大数据情况下的效率问题(初级的就行)。	A: 满足第1--6条; B: 满足1--5条; C: 满足第1--3条; D: 能满足第1--2条; E: 以上均不满足。
解决问题能力	按实践要求合理设计并定义出数据库结构, 根据系统需求构建查询语句, 认真准备实验数据, 记录实验结果, 对实验原理及实验结果分析准确, 归纳总结充分。。	1. 实验数据较多 满足查询和统计要求。 2. 完成了任务书要求的5个存储过程和Java调用。 3. 单独银行转账按月缴费, 有保存转账明细和交款明细。 4. 完成了抄表记录和生成订单的存储过程(或是触发器)。 5. 银行转账和余额是 按月缴费, 有保存转账明细交款明细余额增减记录。 6. 银行转账和余额可以按任意数额缴费, 滞纳金计算正确, 冲正时正确, 有保存转账明细 交款明细 余额增减记录。	A: 满足第1 2 4 6条; B: 满足1 2 4 5 条; C: 满足第1--4条; D: 能满足第1--3条; E: 以上均不满足。
学习能力	按实践要求, 在商用数据库系统平台上, 自学并使用PL/SQL程序代码实现系统要求, 程序结构简洁清晰, 运行结果正确。在解决问题的过程中有自己独到的见解, 并能够有所创新。	1. 变量定义规范 类型选择正确, 最好是改变表中的字段类型后也不影响。 2. 会使用数组保存一个客户的多个设备欠款。 3. 会使用包和包体写能够返回游标的存储过程。 4. 会使用异常机制 来处理事务问题。 5. 会设计底层的工具类存储过程, 方便上面调用和复用。	A: 满足第1--5条; B: 满足1--4条; C: 满足第1--3条; D: 能满足第1--2条; E: 以上均不满足。
报告质量	实践报告格式规范, 报告内容充实、正确, 报告叙述逻辑严密, 可准确反映出实践过程中分析问题和解决问题的过程和结果。	1. 报告内容完整, 叙述清晰准确; 2. 报告格式统一, 图表使用规范; 3. 报告逻辑清晰, 可准确反映系统的设计和实现过程; 4. 报告用词准确, 符合科技文档写作要求。	A. 能满足第1--4条; B. 能满足第1--3条; C. 能满足第1--2条; D. 能满足第1条; E. 以上均不满足。

教师评价:

评价内容	具体要求	分 值	得 分
分析问题能力	按实践课要求，认真分析需求规约，界定好系统需求范围，设计系统的E-R图，E-R图设计合理，能将概念模型转换给关系模式，满足系统需求。	30	
解决问题能力	按实践要求合理设计并定义出数据库结构，根据系统需求构建查询语句,认真准备实验数据，记录实验结果，对实验原理及实验结果分析准确，归纳总结充分。	20	
学习能力	按实践要求，在商用数据库系统平台上，自学并使用PL/SQL程序代码实现系统要求，程序结构简洁清晰，运行结果正确。在解决问题的过程中有自己独到的见解，并能够有所创新。	20	
报告质量	实践报告格式规范，报告内容充实、正确，报告叙述逻辑严密，可准确反映出实践过程中分析问题和解决问题的过程和结果。	30	
总 分			