1. 问题定义

随着互联网时代的到来，人们的生活方式发生了巨大的改变，可以说互联网大大地便利了人们的生活。为了方便普通居民缴纳电费，银行打算和供电所合作推出代缴费服务。在本缴费服务中，要求完成一些和缴费相关的基础功能：

① 查询：

可以根据用户的 ID 来查询到用户的姓名，住址，电话等信息。并且能计算出用户名下所有设备的基础费用，附加费用，违约金。

② 缴费：

针对设备的 ID 进行缴费，在缴费时候需要记录相关的信息如缴费日期，缴费金额银行流水号等。并且要自动计算用户的欠费情况，自动支付一些可以缴纳的费用。（如果缴费没有达到多个月的电费，允许缴纳部分月的电费）

③ 冲正：

当用户输入错了设备号，为其他用户缴费后，需要进行冲正。

④ 对总账：

每天需要进行银行和供电所的对账，首先要对总账。对照银行的缴费记录数和全部缴费金额是否与供电所得缴费记录数和全部缴费金额相同，如果不相同，将异常记录保存。

⑤ 对明细：

如果对总账时候产生了异常，就需要对明细。对明细时候，需要检查是否存在三种异常。第一种是银行与供电所都有某银行流水号的记录，但是钱数不对。第二种是银行遗漏了供电所的缴费记录。第三种是供电所遗漏银行的缴费记录。检查完毕后，将找到的异常保存。

除了以上的基础功能，还需要设计一些用户友好功能。例如制作图形用户界面，方便用户输入数据，增加结果的可读性。除此以外，还需要进行对一些原始数据的处理，例如对一些异常代码进行转化处理，将简单的解释展示给用户，使得用户更加容易理解。

2. 数据库设计

2.1 ER 图的设计

本次数据库的ER图设计经过了两个阶段,首先根据要求先设计了一个初始版本(图 2.1)，在完成初始版本后，通过老师的讲解以及对初始版本的分析，得出了最终的版本（图 2.2)。

在初始版本设计数据库的时候,首先是提取了需求中的一些实体(如供电所用户,设备，抄表记录，账单，供电所记录等)。随后分析了每一个实体所拥有的属性，例如在用户中，首先必须要有主键来标识用户，其次还应该包括用户的一些信息，所以在用户中有 ID（主键)，姓名，地址，电话。除了这些以后，还要考虑到一些属性是与其他表有联系的，例如在设备中,用户 ID 就是外键。在完成这些以后，就要对各个实体之间的关系进行细致分析。因为在初始版本中在充值关系没有处理好，错误的将画流程图的思想引入，这一部分在随后的改进版本中进行阐述。

在最终版本中，对原始版本的缴费部分做了修改，因为缴费关系存在，因此在随后创建表的时候就会创建一张新表来储存供电所缴费记录，不需要创建该实体。除此以外，异常对账的信息虽然是由供电所缴费记录和银行记录所产生，但是在 ER 图中并不需要像画流程图一样对其展现，这层逻辑在随后的实际操作中完成就好。

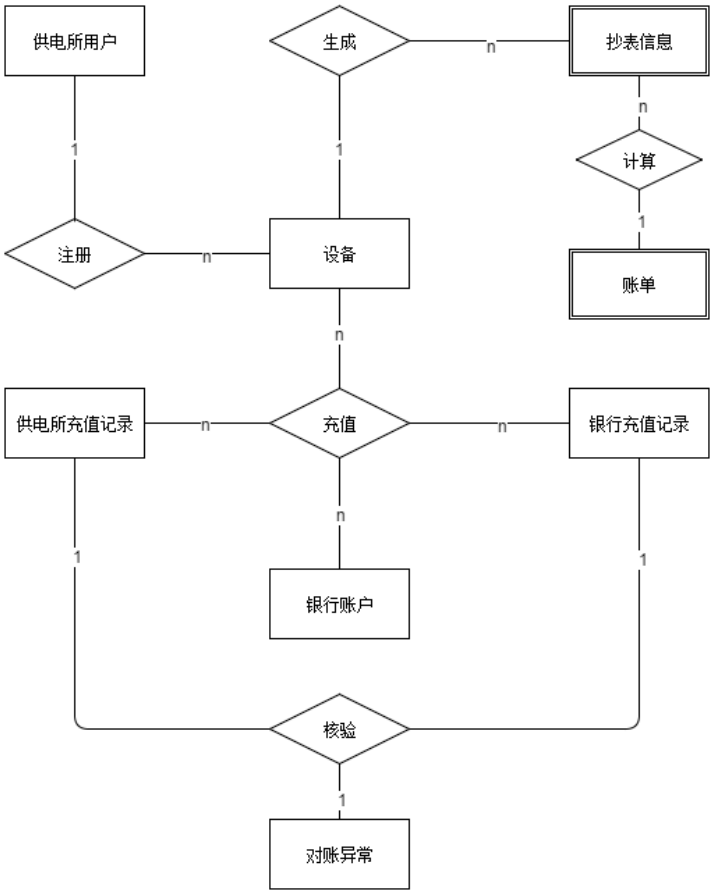

图 2.1 初始版本 ER 图

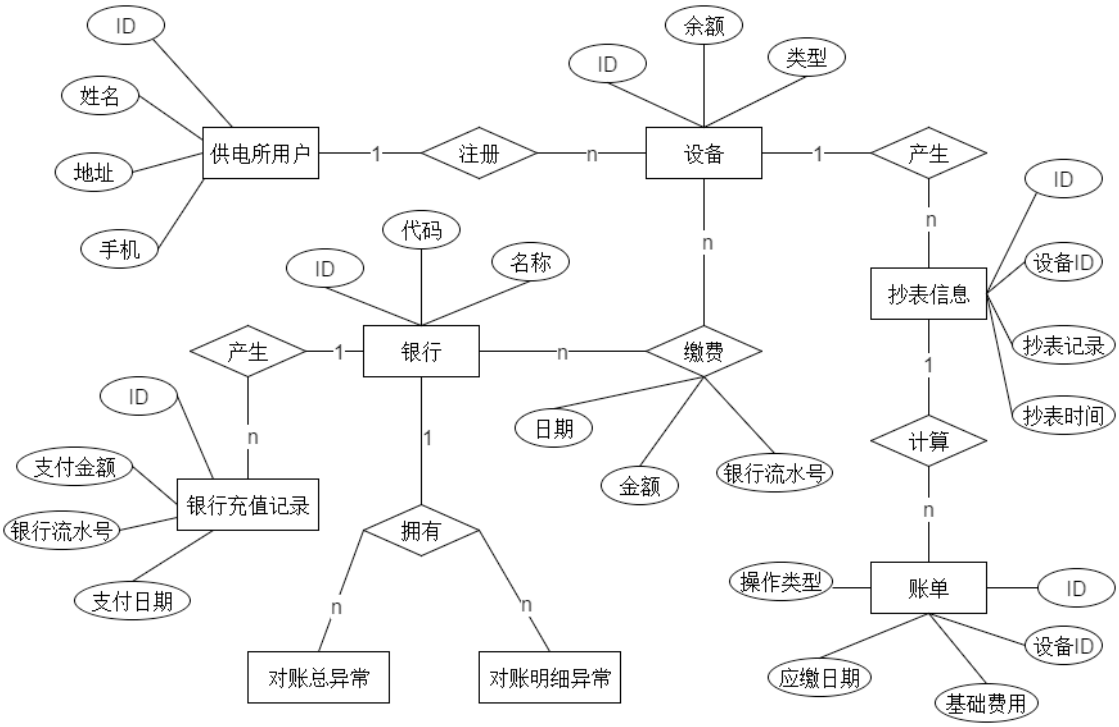

图 2.2 改进版本 ER 图

2.2 ER 图转化成表

在完成 ER 图以后，就要根据 ER 图来创建表，根据“一对一”,“一对多”,“多对多”等关系，可以创建出以下的初始表格。(注：主键有下划线，外键用加粗表示。)

供电所用户(ID,姓名，地址，手机)

设备（ID，**用户 ID**，余额，操作类型）

抄表信息（ID，**设备 ID**，抄表时间，抄表记录）

账单（ID，**设备 ID**，基础费用，应缴日期，操作类型）

银行（ID，银行名，银行代码）

缴费（ID，**银行代码**，**设备号**，缴费金额，缴费日期，缴费类型，银行流水号）

银行充值记录（ID，支付金额，支付日期，银行流水号）

对账总异常(ID,**银行代码**，对账日期，银行总缴费数，供电所总缴费数，银行总账，供电所总账)

对账异常明细(ID，**银行代码**，对账日期，银行钱数，供电所钱数，银行流水号)

2.3 范式化

在有了初始表格后，为了设计出更合理的数据库，还需要进行范式化，来检验这些表格是否满足范式要求（在这里检验的都是 3NF，即属性不可以再分，非主属性完全依赖主键，没有传递依赖）。

因为在这里每张表都引入了 ID，这样方便处理而且这些表结构相对简单，经过检验，都满足 3NF 范式。例如在缴表中，ID 唯一决定一条缴费记录，这里没有用银行流水号来当主键，因为有时候会有冲正的缴费记录，这时候银行流水号就不唯一了。也不存在着传递依赖。

3．数据库端的系统实现

3.1 查询语句

3.1.1 查询出所有欠费用户

涉及到的表：用户，设备，应缴费用表

实现思路：通过用户和设备表可以找到一个用户全部的设备，然后去应缴费用表里面查询这些设备是否欠费，最终找到欠费用户。

实现代码：

```
select distinct client.id, client.name, device.deviceid, device.type
from client join device on (client.id = device.clientid) join receivables on
(device.deviceid = receivables.deviceid)
where receivables.flag = '0';
```

运行结果：

ID	NAME	DEVICEID	TYPE

图 3.1

3.1.2 查询出拥有超过 2 个设备的用户

涉及到的表：用户，设备

实现思路：通过用户和设备表可以找到用户全部的设备，再统计用户设备的数量就可以找到拥有超过 2 个设备的用户。
实现代码：

```
select client.id, client.name
from client join device on (client.id = device.clientid)
group by client.id, client.name having count(device.deviceid) > 2;
```

运行结果：

图 3.2

3.1.3 统计电力企业某每天的总应收费用，实收费用
涉及到的表：缴费表，银行记录表
实现思路：首先在缴费表里面通过日期来统计每天的总应收费用，再在银行记录里面通过日期来统计每天实应收费用，最后通过日期将这两个连接起来。
实现代码：

```
select shoultable.payday, shoulpay,realpay
from (select to_char(payfee.paydate,'yyyy-mm-dd') payday, sum(payfee.paymoney)
shoulpay
      from payfee group by to_char(payfee.paydate,'yyyy-mm-dd')) shoultable --
应该收的费用
      join
      (select to_char(payfee.paydate,'yyyy-mm-dd') payday,
sum(bankrecord.payfee) realpay
      from payfee join bankrecord on(payfee.bankserial = bankrecord.bankserial)
      group by to_char(payfee.paydate,'yyyy-mm-dd')) realtable  --实际收到的费
用
      on (shoultable.payday = realtable.payday) ;
```

运行结果：

	PAYDAY	SHOULPAY	REALPAY
1	2016-08-01	204605	204605
2	2016-09-01	204605	204605

图 3.3

3.1.4 查询出所有欠费超过半年的用户
涉及到的表：用户，设备，应缴费用表
实现思路：因为每月每个设备都会有一条应缴费用记录，一个用户欠费超过了半年，那么就会累计超过 6 条未缴费记录，因此根据这个就可以找到欠费超过半年的用户。
实现代码：
运行结果：

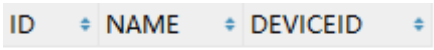

图 3.4

3.1.5 查询任意用户的欠费总额

涉及到的表：用户，设备，应缴费用表

实现思路：在本功能关键是要对用户的设备先进行分类计算（类型 1 设备和类型 2 设备的计费方式不同），然后再进行汇总。

实现代码：

```
select client.id, client.name, sum(owetable.fee) oweallmoney
from client join device on (client.id = device.deviceid) join
  (select device.deviceid, sum(receivables.basicfee * 1.18) fee
  from receivables join device on (receivables.deviceid = device.deviceid)
  where flag = '0' and type = '01'
  group by device.deviceid  --类型 1 设备的欠费表
  union
  select device.deviceid, sum(receivables.basicfee * 1.23) fee
  from receivables join device on (receivables.deviceid = device.deviceid)
  where flag = '0' and type = '02' --类型 2 设备的欠费表
  group by device.deviceid) owetable  --欠费总表
on (device.deviceid = owetable.deviceid)
group by client.id, client.name;
```

实现结果：

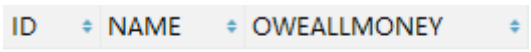

图 3.5

3.1.6 查询出某个月用电量最高的 3 名用户

涉及到的表：用户，设备，抄表记录

实现思路：通过抄表记录可以计算出每个月的用电量（后一个月减去前一个月，这里用 9 月 9 月来测试），然后可以对用电设备排序，最后通过用户设备表可以找到对应的用电量最高的 3 名用户。

实现代码：

```
select client.id, client.name, sum((sunm2-sunm1)) elec
from client join device on (client.id = device.deviceid) join
  (select electricity.deviceid,electricity.yearmonth,electricity.snum sunm1
  from electricity
  where electricity.yearmonth = '201608') electricity1 join
  (select electricity.deviceid,electricity.yearmonth,electricity.snum sunm2
  from electricity
  where electricity.yearmonth = '201609')electricity2 on (electricity1.deviceid
= electricity2.deviceid )
```

```
on (electricity1.deviceid = device.deviceid)
where rownum <= 3
group by client.id, client.name
order by elec desc ;
```

运行结果：

	ID	NAME	ELEC
1	5	张5	50
2	1	张1	50
3	3	张3	50

图 3.6（注：因为全部用户都是 50 随机截取前三个）

3.1.7 查询出电力企业某每个月哪天的缴费人数最多

涉及到的表：缴费表

实现思路：在缴费表中通过月份来分类，再在每个月份中统计每天的缴费人数，最后找到每个月缴费人数最多的那一天。（这里用 8 月来举例子）

实现代码：

```
select payday, paynumber
from (select to_char(payfee.paydate,'yyyy-mm-dd') payday,
count(payfee.bankserial) paynumber
      from payfee
      where to_char(payfee.paydate,'yyyy-mm') = '2016-08'
      group by to_char(payfee.paydate,'yyyy-mm-dd')) paynumbertable
where paynumber = (select max(paynumber) from(select
to_char(payfee.paydate,'yyyy-mm-dd') payday, count(payfee.bankserial) paynumber
from payfee where to_char(payfee.paydate,'yyyy-mm') = '2016-08'
group by to_char(payfee.paydate,'yyyy-mm-dd')));
```

运行结果：

	PAYDAY	PAYNUMBER
1	2016-08-01	1200

图 3.7

3.1.8 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。

涉及到的表：设备表

实现思路：先通过设备型号进行分类统计每个型号的使用人数，最后在一起显示出来。

实现代码：

```
(select device.type ,count(device.deviceid) usenumber
from device
where type = '01'
group by  device.type  --类型 1 设备的使用人数
union
select device.type ,count(device.deviceid) usenumber
from device
where type = '02'
```

```
group by  device.type) order by usenumber desc;
```
运行结果：

	TYPE	USENUMBER
1	02	601
2	01	599

图 3.8

3.1.9 统计每个月各银行缴费人次，从高到低排序。
涉及到的表：银行表，缴费表
实现思路：先将银行和各个缴费记录对应，然后通过缴费的月份来统计缴费人数，最后按照从高到低的顺序排序。
实现代码：
```
select bank.code,bank.name,to_char(payfee.paydate,'yyyy-mm')
paymonth,count(bankrecord.payfee) paynumber
from bank join  bankrecord on(bank.code = bankrecord.bankcode) join payfee
on(bankrecord.bankserial = payfee.bankserial)
group by bank.code,bank.name,to_char(payfee.paydate,'yyyy-mm')
order by paynumber desc;
```
运行结果：

	CODE	NAME	PAYMONTH	PAYNUMBER
1	19	招商银行	2016-08	1200
2	19	招商银行	2016-09	1200

图 3.9

3.1.10 查询出电力企业所有新增用户（使用设备不足半年）
涉及到的表：用户，设备，应缴费用表
实现思路：因为每个设备每个月都会有 1 条应缴费用记录，如果有一个设备的应缴费记录少于 6 条的话，证明该设备使用不足半年，然后通过用户，设备表即可找到对应的用户。
实现代码：
```
select client.id,client.name,device.deviceid,usetimetable.usemonth
from client join  device on (client.id = device.clientid) join
  (select receivables.deviceid, count(receivables.deviceid) usemonth
   from receivables group by receivables.deviceid) usetimetable --每个设备的使用月数
on (device.deviceid = usetimetable.deviceid)
where usetimetable.usemonth < 6;
```
运行结果：

	ID	NAME	DEVICEID	USEMONTH
1	126	张126	126	2
2	128	张128	128	2
3	130	张130	130	2
4	132	张132	132	2
5	134	张134	134	2
6	136	张136	136	2

图 3.10（注：全部的用户都少于半年，这里截取部分）

3.2 存储过程

3.2.1 查询

实现思路：首先是找到用户全部的设备，然后根据用户的设备类型分类统计（不同类型的收费标准不同），在类型 1 中，因为是不管跨不跨年违约金计算的方法都是相同的，因此只要根据系统时间来计算。在类型 2 中，欠费跨年和欠费不跨年的计算方式不同，所以得判断设备是否跨年，然后再分类进行计算。最后再将欠费总和。

实现关键代码：

```
-获取用户的设备费用
  for r_devicefee in mycursor loop
    v_paydate := to_date(r_devicefee.yearmonth,'YYYYMM');
    if(r_devicefee.type = '01') then --如果是家用设备

      --计算基础费用
      v_basicfee := v_basicfee + r_devicefee.basicfee;
      v_extrafee := v_extrafee + r_devicefee.basicfee* 0.18;
      v_fee := r_devicefee.basicfee * 1.18;
      --计算违约天数 应缴
      v_owedays := floor(sysdate - add_months(v_paydate,1));
      --计算应缴费
      v_penalty := v_penalty  + v_fee * v_owedays / 1000;
      v_payfee := v_payfee + v_fee + v_fee * v_owedays / 1000;
    elsif(r_devicefee.type = '02') then --如果不是家用设备
      --计算基础费用
      v_basicfee := v_basicfee + r_devicefee.basicfee;
      v_extrafee := v_extrafee + r_devicefee.basicfee* 0.25;
      v_fee := r_devicefee.basicfee * 1.25;
      --计算违约金部分 这里分为两部分 跨年和不跨年
      if(months_between(sysdate,v_paydate)/12 <= 1) then --没有跨年
         --计算违约天数
         v_owedays := floor(sysdate - add_months(v_paydate,1));
         --计算违约金
         v_penalty := v_penalty  + v_fee * v_owedays / 500;
         --计算应缴费
         v_payfee := v_payfee + v_fee + v_fee * v_owedays / 500;
      else --超过一年
         --计算违约天数 欠费当年
```

```
        v_owedays := floor(last_day(add_months(trunc(v_paydate,'y'),11)) -
v_paydate);
        --计算违约金 欠费当年
        v_payfee := v_payfee + v_fee * v_owedays / 500;
        v_penalty := v_penalty + v_fee * v_owedays / 500;
        --计算违约天数 欠费次年以后
        v_owedays := floor(sysdate - add_months(trunc(v_paydate,'yyyy'),12));
        --计算违约金 欠费次年以后
        v_penalty := v_penalty + v_fee * v_owedays * (3/1000);
        v_payfee := v_payfee +  v_fee * v_owedays * (3/1000);
        --计算应缴费
        v_payfee := v_payfee + v_fee;
```

3.2.2 缴费

实现思路：在缴费表中插入的记录，在此要注意的是对异常情况的处理，如果用户给不存在的设备缴费要做出相应的处理。

实现关键代码：

```
create or replace procedure p_pay
 (v_deviceid in device.deviceid%type,
  v_paymoney in payfee.paymoney%type,
  v_payresult out number
  )
is
  v_number payfee.id%type := 0; -- 记录现在表中全部数据的数据 方便生成 id
begin
  select count(*) into v_number from payfee;
  insert into payfee
values(v_number+1,v_deviceid,v_paymoney,sysdate,19,2001,'ZS'||to_char(sysdate,'
YYYYMMDD')||lpad((v_number+1),5,0));
  v_payresult := 1;-- 1 代表成功 0 代表失败
  commit;
  exception
    when others then
    v_payresult := 0;
end;
```

3.2.3 冲正

实现思路：对已经存在的缴费记录进行冲正，根据银行流水号，将冲正金额设置为缴费金额的相反数，并且设置相应的操作类型。

实现关键代码：

```
create or replace procedure p_correct
 (v_bankserial in payfee.bankserial%type,
  v_result out number
  )
```

```
is
  v_number payfee.id%type := 0; -- 记录现在表中全部数据的数据 方便生成 id
  t_payfee payfee%rowtype;
begin
  select count(*) into v_number from payfee;
  select * into t_payfee from payfee where payfee.bankserial = v_bankserial;
  insert into payfee values(v_number+1,t_payfee.deviceid,'-
'||t_payfee.paymoney,sysdate,19,2002,v_bankserial);
  v_result := 1;
  commit;
  exception
    when others then
    v_result := 0;
end;
```

3.2.4 对总账

实现思路：根据日期计算某一银行某天全部的缴费数和缴费金额，然后比较供电所缴费记录里面的缴费数和缴费金额，不相同的话就往异常表里边添加一条记录。

实现关键代码：

```
--根据日期来选取制定的银行支付记录
   cursor mycursor_bankrecord is select * from bankrecord
   where bankcode = v_bankcode and substr(bankserial,3,8) =
to_char(v_checkdate,'YYYYMMDD');

   cursor mycursor_payfee is select* from payfee
   where to_char(paydate,'YYYYMMDD') = to_char(v_checkdate,'YYYYMMDD');
   t_bankrecord bankrecord%rowtype;
   t_payfee payfee%rowtype;
   v_number number := 0;
begin
   v_totalcount := 0;
   v_totalmoney := 0;
   v_ourtotalmoney := 0;
   v_ourtotalcount := 0;
   --计算银行的总账单数和总钱数
   for t_bankrecord in mycursor_bankrecord loop
     v_totalcount := v_totalcount + 1;
     v_totalmoney := v_totalmoney + t_bankrecord.payfee;
   end loop;
   --计算供电所的总账单数和总钱数
   for t_payfee in mycursor_payfee loop
     if(t_payfee.type = '2001') then
       v_ourtotalcount := v_ourtotalcount + 1;
       v_ourtotalmoney := v_ourtotalmoney + to_number(t_payfee.paymoney);
```

```
    elsif(t_payfee.type = '2002') then
      v_ourtotalcount := v_ourtotalcount - 1;
      v_ourtotalmoney := v_ourtotalmoney + to_number(t_payfee.paymoney);
    end if;
  end loop;
  --计算记录表中的数目方便创建 id
  select count(*) into v_number from checkresult;
  --将这些记录插入到表中
  insert into checkresult values(v_number+1,v_checkdate,v_bankcode,
              v_totalcount,v_totalmoney, v_ourtotalcount, v_ourtotalmoney);
```

3.2.5 对明细

实现思路：当对明细账的时候，需要检测三种情况。银行记录和供电所记录存在相同流水号的记录，但是金额不符。银行遗漏记录和供电所遗漏记录。

实现关键代码：

```
--先检测对上流水号的细节
  for t_combine in mycursor_combine loop
    select count(*) into v_number from check_exception;
    if(to_char(t_combine.payfee,'9999999.99') <>
to_char(t_combine.paymoney,'9999999.99'))then
      insert into check_exception
values(v_number+1,v_checkdate,t_combine.bankcode,t_combine.bankserial,t_combine
.payfee,t_combine.paymoney,'001');
    end if;
  end loop;

  --检测供电所有没有遗漏银行的记录
  for t_payfee in mycursor_our loop
      select count(*) into v_existnumber from bankrecord
      where bankrecord.bankserial = t_payfee.bankserial;

      if(v_existnumber = 0) then
        select count(*) into v_number from check_exception;
        insert into check_exception
values(v_number+1,v_checkdate,t_payfee.bankcode,t_payfee.bankserial,0,t_payfee.
paymoney,'002');
      end if;
  end loop;
  --检测银行有没有遗漏供电所的记录
  for t_bankrecord in mycursor_bank loop
    select count(*) into v_existnumber from payfee
    where payfee.bankserial = t_bankrecord.bankserial;
    if(v_existnumber = 0) then
       select count(*) into v_number from check_exception;
```

```
       insert into check_exception
values(v_number+1,v_checkdate,t_bankrecord.bankcode,t_bankrecord.bankserial,t_b
ankrecord.payfee,0,'003');
    end if;
  end loop;
```

3.2.6 更新

实现思路：当一天核对完信息以后，可以根据缴费记录来更新账户的余额，先将用户的全部缴费充值到设备余额上。当充值完毕以后，会自动判断有没有欠费记录，如果有的话来判断是否能够支付，如果可以支付，就自动付清。

实现关键代码：

```
  --将充值加到设备余额上
  for r_payfee in mycursor_payfee loop
    update device set balance = balance + to_number(r_payfee.paymoney);
  end loop;
  --检测需要交费的表 如果遇到需要交费的并且余额够的话 就交费
  for r_receivables in mycursor_receivables loop
    if(r_receivables.flag = '0') then -----需要交费
      select balance,device.type into v_balance,v_devicetype from device where
device.deviceid = r_receivables.deviceid;
      --计算违约金
      if(v_devicetype = '01') then --如果是家用设备
        --计算基础费用
        v_fee := r_receivables.basicfee * 1.18;
        --计算违约天数 应缴
        v_owedays := floor(sysdate - add_months(v_paydate,1));
        --计算应缴费
        v_fee := v_fee + v_fee * v_owedays / 1000;
      elsif(v_devicetype = '02') then --如果不是家用设备
        --计算基础费用
        v_fee := r_receivables.basicfee * 1.25;
        --计算违约金部分 这里分为两部分 跨年和不跨年
        if(months_between(sysdate,v_paydate)/12 <= 1) then --没有跨年
           --计算违约天数
           v_owedays := floor(sysdate - add_months(v_paydate,1));
           --计算应缴费
           v_fee :=  v_fee + v_fee * v_owedays / 500;
        else --超过一年
           --计算违约天数 欠费当年
           v_owedays := floor(last_day(add_months(trunc(v_paydate,'y'),11)) -
v_paydate);
           --计算违约金 欠费当年
           v_fee := v_fee + v_fee * v_owedays / 500;
           --计算违约天数 欠费次年以后
```

```
          v_owedays := floor(sysdate -
add_months(trunc(v_paydate,'yyyy'),12));
          --计算违约金 欠费次年以后
          v_fee := v_fee +  r_receivables.basicfee * 1.25 * v_owedays *
(3/1000);
        end if;
      end if;
      --计算是否够缴费 如果余额够的话就进行缴费
      if(v_balance >= v_fee) then
        update receivables set flag = '1';
        update device set balance = v_balance - v_fee where deviceid =
r_receivables.deviceid;
      end if;
    end if;
  end loop;
```

4．程序实现

在本系统中，首先是在数据库中构造了一些存储过程。随后用 Java Swing 创建了图形用户界面，来获取用户的一些输入，然后通过 JDBC 建立与数据库之间的连接，获取到想要的结果后返回，再通过图形用户界面来展示，除此以外，为了增加程序的健壮性，还添加了容错处理，当用户输入错误的数据类型的时候会给出警告，即使数据类型正确，在数据库中没有对应的消息时候，也会给出提醒。

在此展示一段简单的连接数据库代码：（实现用户查询）

```
//获取用户 ID
String userID = mainGUI.getUserIDTextField().getText();
//建立数据库连接
Connection connection = getDatabaseConnection();//这是获取数据库连接的自定义方法 连接指定数据库 返回连接
//执行存储过程
CallableStatement s = connection.prepareCall("{call p_query(?,?,?,?,?,?,?,?)}");
//设置参数值
s.setInt(1,Integer.parseInt(userID));
s.registerOutParameter(2,Types.VARCHAR);
s.registerOutParameter(3,Types.VARCHAR);
s.registerOutParameter(4,Types.DOUBLE);
s.registerOutParameter(5,Types.DOUBLE);
s.registerOutParameter(6,Types.DOUBLE);
s.registerOutParameter(7,Types.DOUBLE);
s.registerOutParameter(8,Types.VARCHAR);
//执行查询
s.execute();
//获取需要的值
String username = s.getString(2); //用户名
```

```
String address = s.getString(3); //用户地址
double basicFee = s.getDouble(4);//基础费用
double extraFee = s.getDouble(5);//额外费用
double penalty = s.getDouble(6);//违约金
double totalPay = s.getDouble(7);//总应支付
String telNumber = s.getString(8);//电话号码

//执行缴费记录的 sql
Statement statement = connection.createStatement();
//获取行数
String sql1 = "select count(*) " +
        "from device join receivables on (device.deviceid = receivables.deviceid) " +
        "where device.clientid = " + userID + "and flag = '0'";
ResultSet rs1 = statement.executeQuery(sql1);
int rowNumber = 0;
while (rs1.next()){
    rowNumber = rs1.getInt(1);
}
//查询内容
String sql2 = "select device.deviceid, receivables.yearmonth, receivables.basicfee " +
            "from device join receivables on (device.deviceid = receivables.deviceid) " +
            "where device.clientid = " + userID + "and flag = '0'";
ResultSet rs2 = statement.executeQuery(sql2);
//获取列数目
int columnNumber = rs2.getMetaData().getColumnCount();
String [][]results = new String[rowNumber][columnNumber];
int i=0;
while (rs2.next()) {
    for (int j = 1; j <= columnNumber; j++) {
        results[i][j-1] = rs2.getString(j);
        if(j==2){ //修改日期格式
            results[i][j-1] = new StringBuffer(results[i][j-1]).insert(4," 年 ").append(" 月
").toString();
        }
    }
    i++;
}
connection.close();

//更新界面
mainGUI.getNameLabel().setText("用户名：" + username);
mainGUI.getTelLabel().setText("手机："+telNumber);
mainGUI.getAddressLabel().setText("地址：" + address);
mainGUI.getBasicfeeLabel().setText("基本费用："+ basicFee);
```

```
mainGUI.getExtrafeeLabel().setText("额外费用：" + extraFee);
mainGUI.getPenaltyLabel().setText("违约金：" + penalty);
mainGUI.getTotalLabel().setText("总费用：" + totalPay);
String[] tableTitle = {"设备号","应缴月份","基础费用"};
mainGUI.getFeeScrollPane().setViewportView(new JTable(results,tableTitle));
```

4.1 查询：

图 4.1 查询界面

4.2 缴费及冲正：

图 4.2 缴费及冲正界面

4.3 对账界面：

图 4.3 对账及冲正页面

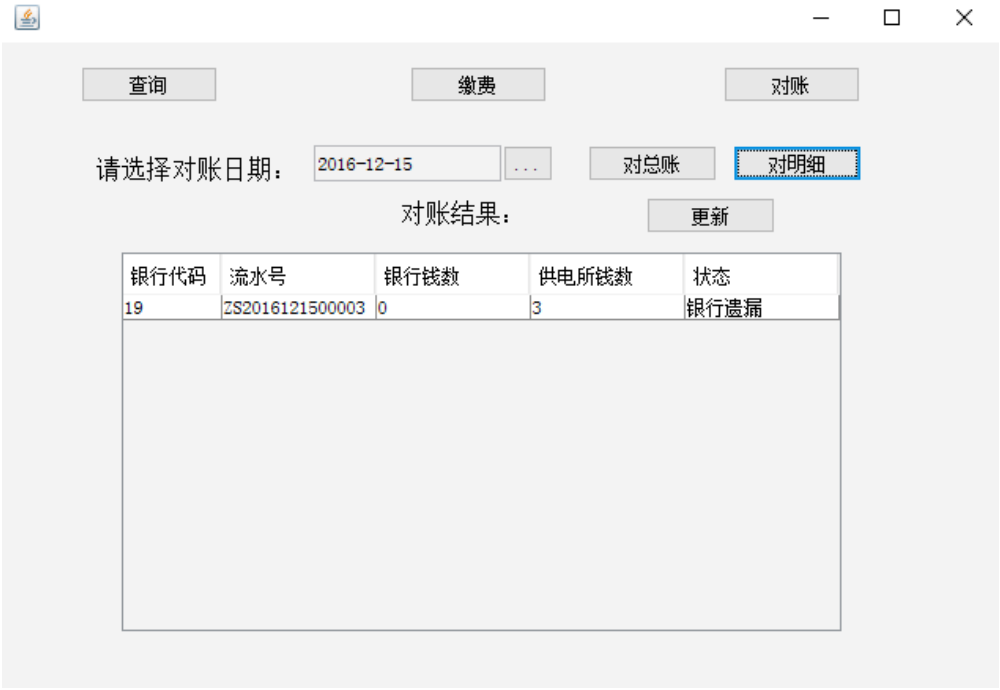

图 4.4 对账明细

4.4 容错处理：

图 4.5 错误输入类型容错

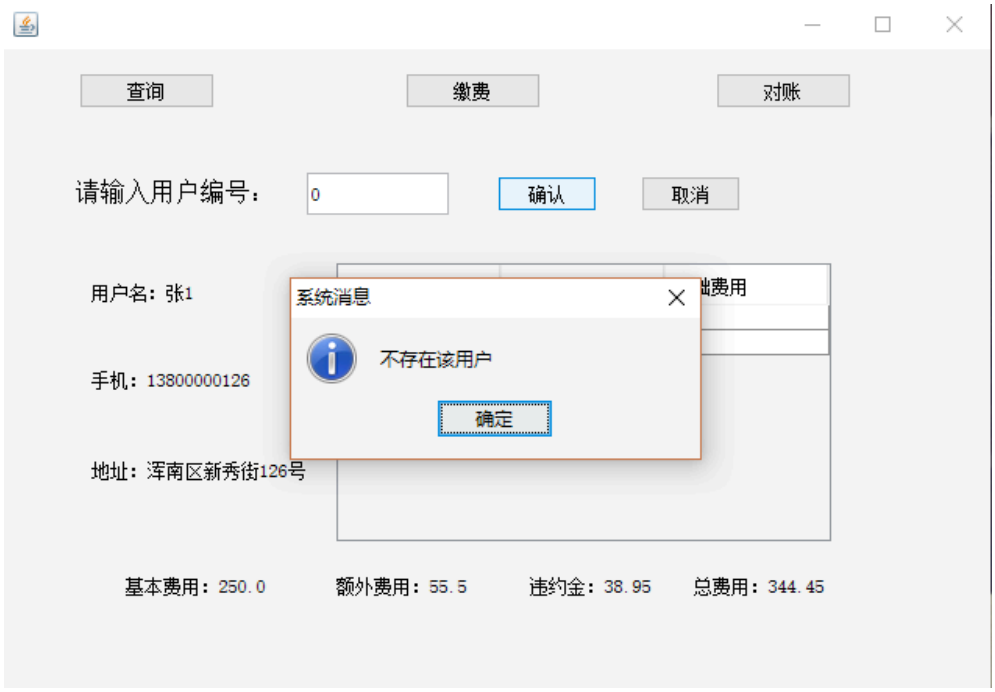

图 4.6 数据库错误容错

5. 遇到的问题及其解决方案

5.1 最初设计的 E-R 图不够恰当

在最初设计 ER 图的时候，错误的引入了画流程图的思想，导致画出来的 ER 图有部分过于复杂。

解决方案：在经过老师的讲解以及与同学的交流，对原有的 ER 图进行了修改，改正了

一个关系过于复杂，并且删去了一些不必要的逻辑。

5.2 对存储过程不熟悉，得不到预期结果

由于以前没有接触过存储过程，在最初编写的时候，总是会产生一些错误，例如函数调用错误，语法错误，变量声明错误等。

解决方案：发现错误较多的时候，先是停止了编写。先把老师发的教案复习了一遍，然后上网查询了一些教学视频，认真学习了存储过程。

5.3 使用 Java 语言与数据库连接发生错误

因为本次实验使用了 Java 语言与数据连接，并调用存储过程，在两者之间实现数据传输。最初调试的时候由于缺乏经验，不清楚两者之间数据类型的对照，总是传入错误的数据类型，常常执行失败。

解决方案： 在意识到这一错误以后，首先查询了一些资料，明白了一些对应的数据类型。并且在 java 端做了一些测试，输入一些值，并将其打印出来，当看到预期的数据格式时，再进行下一步。

6．创新点

6.1 创建用户图形界面

众所周知，使用命令行调用数据库不仅麻烦而且数据展现效果可读性差。因此，本程序编写了用户图形界面，方便用户的输入和辨识数据，提高了系统的使用效率。

6.2 对数据库的查询结果进行二次处理

有时候数据库返回的值不是十分易懂的，比如在对细账的时候，返回的异常代码不易读懂，程序在此基础上进行了改进，根据不同的异常代码翻译出对应的异常类型，增加程序的易用性。

6.3 进行容错处理

在完成一个程序以后，容错处理往往是最容易忽略的，但是这对程序又是至关重要的。本程序检测了数据的合法性以及在数据库中的存在性，使系统变得更加健壮。

7．总结

在这次数据库实验中，以供电所代收费系统为实现对象，首先完成了对数据库的设计（绘制 ER 图等）以及实现（编写建表语句），随后构建了查询语句和存储过程，来实现一些需求的功能。在完成数据库端的构建后，又尝试使用高级语言与数据库建立连接（JDBC），实现数据的交互。并在此基础上使用 Java Swing 来制作图形用户界面，使得用户与数据库的交互更加方便。

除了学习到许多数据库相关的知识，交流能力，解决问题，快速学习能力也得到了提升。在实验开始阶段，有许多需求不清晰，在与老师的交流中获取到了许多有用的信息，也学习到了很多成熟的分析问题的方法。在实现阶段也不是一帆风顺的，这就要求能够快速发现问题，并且通过多种途径去解决问题。在最后的存储过程部分，即使以前没有接触过相关的知识，也在老师的引导下快速的对其进行了掌握并应用到实践中去，最终很好的实现了这一系统。

参考文献

[1] 陈昊鹏，Java 编程思想（第 4 版）[thinking in java] [M]. 机械工业出版社 出版时间: 2007-06-01

[2] 博利厄，SQL 学习指南（第 2 版 修订版））人民邮电出版社 [M]. 出版时间: 2015-02-01