

课程编号: C0801207050

数据库应用程序设计实践 报告



姓 名	许瑞文	学 号	20165086
班 级	软英 1602	指 导 教 师	宋波
开 设 学 期	2017-2018 第 二 学 期		
开 设 时 间	第 25 周 —— 第 26 周		
报 告 日 期	2018.8.31		
评 定 成 绩		评 定 人	
		评 定 日 期	

东北大学软件学院

1. 问题定义

1.1 系统概述

随着电力行业的快速发展，电力公司力求为用户提供更多更便利的服务。在当前移动支付的浪潮下，能让用户“足不出户”即可实现在线缴费是电力公司的迫切需求。但是电力公司因为客户众多且交易量庞大，与银行的交互难免会有错误发生，这就要求我们专门为银行设计一套银行代收费系统，更贴合电力行业的行业属性。

1.2 关键业务

银行代收费系统是我们为电力公司所开发的一套收费管理系统，电力公司通过这套系统可以方便用户通过网银支付电费。实现整个流程的关键业务包括：

(1) 抄表

抄表员每月抄表，系统管理员把抄表记录录入系统，抄表记录包括当前电表数、抄表日期、抄表人等信息，根据抄表记录，系统自动计算每个计费设备当月的应收电费。每个计费设备有唯一编号。

(2) 查询用户欠费金额

用户可以随时查询自己的欠费金额。一个用户名下可有多个计费设备，查询欠费清单时，可查询所有计费设备欠费总和，也可查询单个计费设备欠费金额。

(3) 缴费

在当月电费清单生成完毕后，用户可进行电费缴纳，缴纳金额可是任意金额。剩余金额将被存入用户余额中。

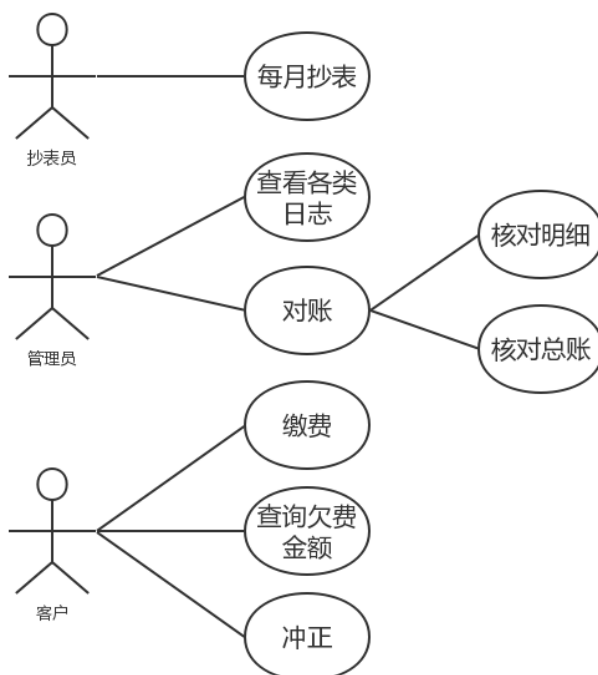
(4) 冲正

用户难免会因为误操作而将钱缴费给了其他用户的设备。用户在缴费过程中如果给其他用户缴费了，在当日 0 点前可以冲正，即把钱收回，放入余额，过了 0 点缴费生效，不能冲正。

(5) 对账

每个银行每日凌晨给电力公司的代缴费系统发送对账信息，代缴费系统记录对账结果，对账明细，对账异常信息。

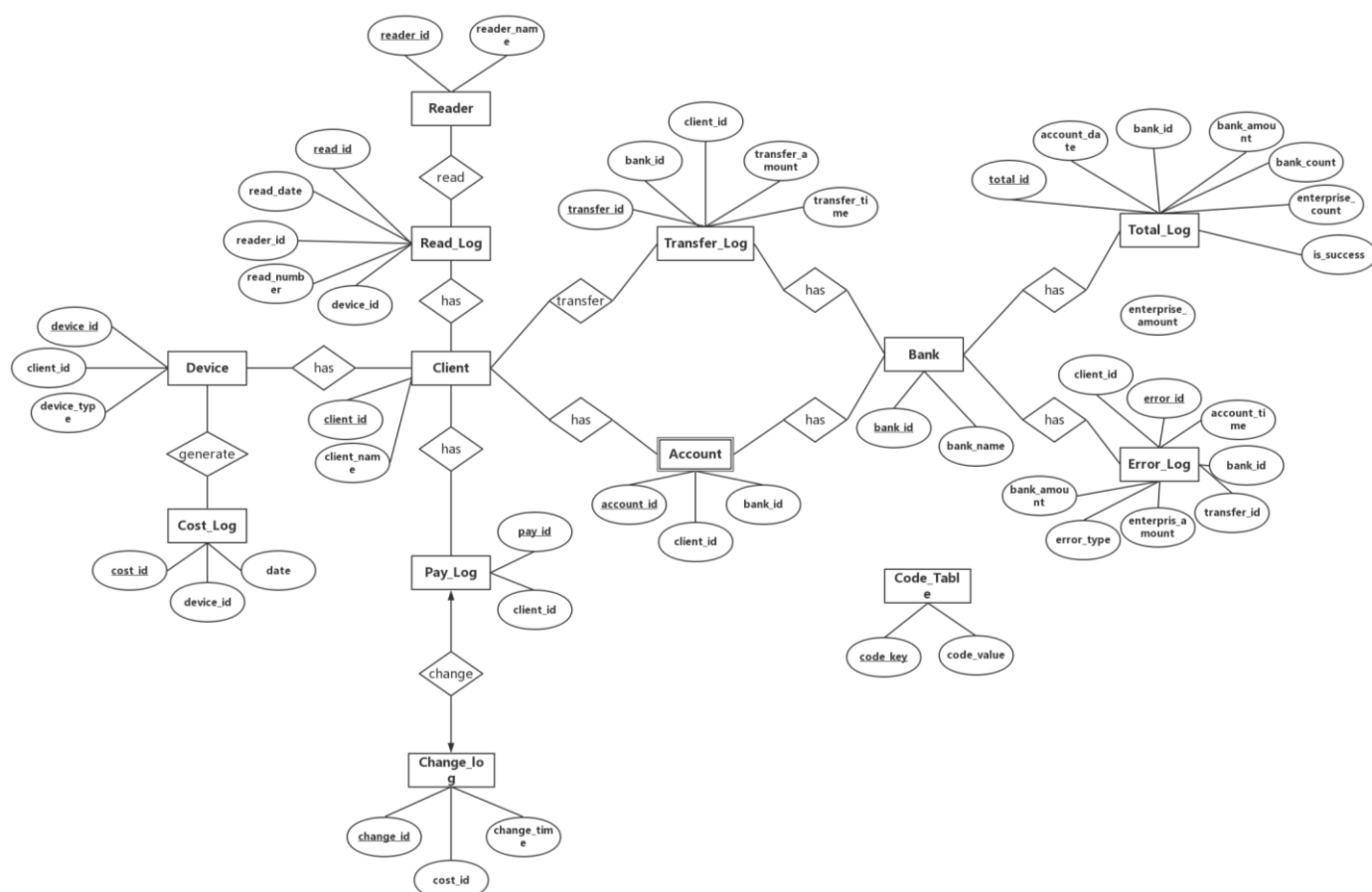
1.3 用例图



2. 数据库设计

2.1 概念结构设计——ER 图

2.1.1 ER 图



共 14 个实体，11 个联系

2.1.2 ER 图所包含的表结构

(一) 实体

(1) account（账户，用于实现单用户多银行卡支付）

名	类型	长度	小数点	不是 null	键
account_id	BIGINT	20	0	√	P.K.
client_id	BIGINT	20	0		F.K.
bank_id	VARCHAR	9	2		F.K.

(2) balance_log（用户的余额变动记录）

名	类型	长度	小数点	不是 null	键
balance_id	BIGINT	20	0	√	P.K.
now_balance	DECIMAL	9	2		
client_id	BIGINT	20	0	√	F.K.
action	VARCHAR	9	0		
date	DATETIME	0	0	√	

(3) bank（银行，用于单用户多银行卡缴费）

名	类型	长度	小数点	不是 null	键
bank_id	BIGINT	9	0	√	P.K.
bank_name	DECIMAL	30	0	√	

(4) change_log (账单改变记录, 用于冲正)

名	类型	长度	小数点	不是 null	键
change_id	BIGINT	20	0	√	P.K.
cost_id	BIGINT	20	0		F.K.
actual_fee_1	DECIMAL	9	2	√	
late_fee_1	DECIMAL	9	2		
pay_date_1	DATETIME	0	0	√	
already_fee_1	DECIMAL	9	2		
pay_state_1	VARCHAR	9	0		
change_time	DATETIME	0	0		

(5) client (用户)

名	类型	长度	小数点	不是 null	键
client_id	BIGINT	20	0	√	P.K.
client_name	VARCHAR	30	0	√	
address	VARCHAR	90	0	√	
balance	DECIMAL	9	2		

(6) code_table (固定信息表)

名	类型	长度	小数点	不是 null	键
code_key	VARCHAR	30	0	√	P.K.
code_value	VARCHAR	9	0	√	

(7) cost_log (账单)

名	类型	长度	小数点	不是 null	键
cost_id	BIGINT	20	0	√	P.K.
device_id	BIGINT	20	0	√	F.K.
date	DATETIME	0	0	√	
begin_number	BIGINT	20	0		
end_number	BIGINT	20	0		
basic_cost	DECIMAL	9	2		
additional_cost_1	DECIMAL	9	2		
additional_cost_2	DECIMAL	9	2		
paid_fee	DECIMAL	9	2		
actual_fee	DECIMAL	9	2		
late_fee	DECIMAL	9	2		
payable_date	DATETIME	0	0		
pay_date	DATETIME	0	0		
already_fee	DECIMAL	9	2		
pay_state	VARCHAR	9	0		

(8) device (设备)

名	类型	长度	小数点	不是 null	键
device_id	BIGINT	20	0	√	P.K.
client_id	BIGINT	20	0		F.K.
device_type	DECIMAL	9	2	√	

(9) error_log (核对明细错误记录)

名	类型	长度	小数点	不是 null	键
error_id	BIGINT	20	0	√	P.K.
account_time	DATETIME	0	0	√	
bank_id	VARCHAR	9	0	√	F.K.
transfer_id	BIGINT	20	0	√	F.K.
client_id	BIGINT	20	0	√	F.K.
bank_amount	DECIMAL	9	2	√	
enterprise_amount	DECIMAL	9	2	√	
error_type	VARCHAR	20	0		

(10) pay_log (缴费记录)

名	类型	长度	小数点	不是 null	键
pay_id	BIGINT	20	0	√	P.K.
client_id	BIGINT	20	0	√	F.K.
device_id	BIGINT	20	0	√	F.K.
pay_time	DATETIME	0	0	√	
pay_amount	DECIMAL	9	2	√	
pay_type	VARCHAR	9	0	√	
bank_id	VARCHAR	9	0		F.K.
transfer_id	BIGINT	20	0		F.K.

(11) read_log (抄表记录)

名	类型	长度	小数点	不是 null	键
read_id	BIGINT	20	0	√	P.K.
read_date	DATETIME	0	0	√	.
device_id	BIGINT	20	0	√	F.K.
read_number	BIGINT	20	0	√	
reader_id	BIGINT	20	0		F.K.

(12) reader (抄表员)

名	类型	长度	小数点	不是 null	键
reader_id	BIGINT	20	0	√	P.K.
reader_name	VARCHAR	30	0	√	.

(13) total_log (核对总账记录)

名	类型	长度	小数点	不是 null	键
total_id	BIGINT	20	0	√	P.K.
account_date	DATETIME	0	0	√	
bank_id	VARCHAR	9	0	√	F.K.
bank_count	BIGINT	20	0	√	
bank_amount	DECIMAL	9	2	√	
enterprise_count	BIGINT	20	0	√	
enterprise_amount	DECIMAL	9	2	√	
is_success	VARCHAR	2	0		

(14) transfer_log (银行交易流水记录)

名	类型	长度	小数点	不是 null	键
transfer_id	BIGINT	20	0	√	P.K.
bank_id	VARCHAR	9	0	√	F.K.
client_id	BIGINT	20	0	√	F.K.

transfer_amount	DECIMAL	9	2	√	
transfer_time	DATETIME	0	0	√	

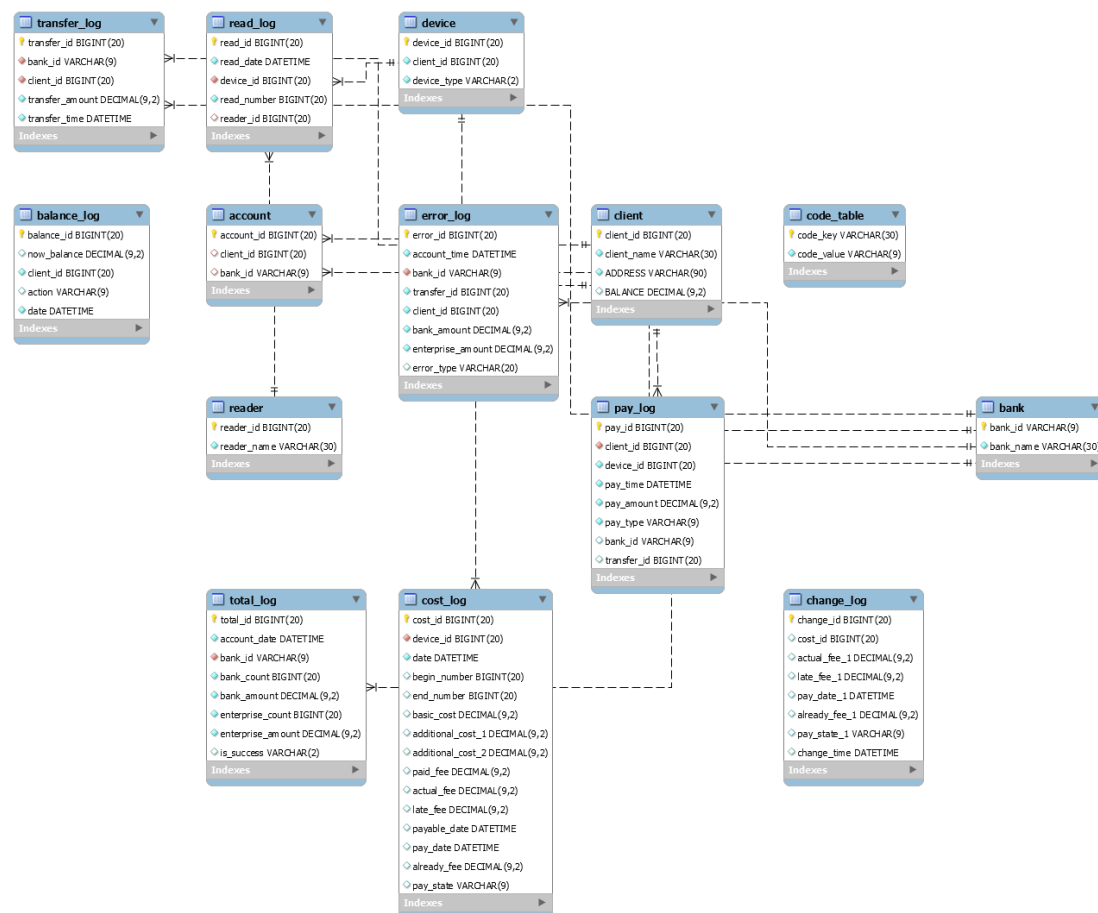
(二) 联系

联系名称	实体一	实体二	说明
generate	device	cost_log	设备对应账单
has	client	device	用户拥有设备
has	client	pay_log	用户缴费生成缴费记录
has	client	read_log	用户拥有抄表记录
read	reader	read_log	抄表员抄表生成抄表记录
change	pay_log	change_log	缴费记录变动时生成改变记录
transfer	client	transfer_log	用户缴费时也生成交易记录
has	client	account	用户拥有多张银行卡
has	bank	account	银行管理自己的用户
has	bank	transfer_log	银行拥有转账流水记录
has	bank	total_log	银行产生总账记录
has	bank	error_log	银行产生明细错误记录

2.2 逻辑结构设计

2.2.1 EER 图

根据概念设计中我们绘制的 ER 图，可以产生 EER 图，便于我们更直观了解各个表真正拥有哪些属性。



2.2.2 将 ER 图映射成关系

利用映射算法，我们将上面我们所设计的 ER 模型转换为关系数据模型。

- (1) account(account_id, client_id, bank_id);
PK(account_id)
FK(bank_id) → bank(bank_id)
- (2) balance_log(balance_id, now_balance, client_id, action, date)
PK(balance_id)
FK(client_id) → client(client_id)
- (3) bank(bank_id, bank_name)
PK(bank_id)
- (4) change_log(change_id, cost_id, actual_fee_1, late_fee_1, pay_date_1, already_fee_1, pay_state_1, change_time)
PK(change_id)
FK(cost_id) → cost_log(cost_id)
- (5) client(client_id, client_name, address, balance)
PK(client_id)
- (6) code_table(code_key, code_value)
PK(code_key)
- (7) cost_log(cost_id, device_id, date, begin_number, end_number, basic_cost, additional_cost_1, additional_cost_2, paid_fee, actual_fee, late_fee, payable_date, pay_date, already_fee, pay_state)
PK(cost_id)
FK(device_id) → device(device_id)
- (8) device(device_id, client_id, device_type)
PK(device_id)
FK(client_id) → client(client_id)
- (9) error_log(error_id, account_time, bank_id, transfer_id, client_id, bank_amount, enterprise_amount, error_type)
PK(error_id)
FK(bank_id) → bank(bank_id)
FK(transfer_id) → transfer(transfer_id)
FK(client_id) → client(client_id)
- (10) pay_log(pay_id, client_id, device_id, pay_time, pay_amount, pay_type, bank_id, transfer_id)
PK(pay_id)
FK(client_id) → client(client_id)
FK(device_id) → device(device_id)
FK(transfer_id) → transfer_log(transfer_id)
- (11) read_log(read_id, read_date, device_id, read_number, reader_id)
PK(read_id)
FK(device_id) → device(device_id)
FK(reader_id) → reader(reader_id)
- (12) reader(reader_id, reader_name)
PK(reader_id)
- (13) total_log(total_id, account_date, bank_id, bank_count, bank_amount, enterprise_count, enterprise_amount, is_success)

PK(total_id)

FK(bank_id) → bank(bank_id)

(14) transfer_log(transfer_id, bank_id, client_id, transfer_amount, transfer_time)

PK(transfer_id)

FK(bank_id) → bank(bank_id)

FK(client_id) → client(client_id)

2.2.3 规范化理论检验

从 2.2.2 的转化得到的关系结果来看，全部 14 个表内都不存在属性依赖于非主属性，所以我们的数据库设计满足 3NF，是比较成功的设计。

3. 数据库端的系统实现

3.1 基础功能实现

3.1.1 抄表同时生成电费清单

(1) 详细需求

抄表员每月抄表，系统管理员把抄表记录录入系统，根据抄表记录，系统自动计算每个计费设备当月的应收电费。抄表记录包括电力设备唯一编号、当前度数、抄表日期、抄表人。

抄表记录被录入系统后，每一条抄表记录在数据库内有一条唯一的记录。有新的抄表记录后，系统根据电费计算规则自动生成电费清单。电费计算规则：

每月电费应缴金额的计算方式：基本费用（电数*每度电金额）+附加费用 1+附加费用 2

附加费用 1：为基本费用的 8%

附加费用 2：计费设备为 01 时为基本费用 10%，计费设备为 02 时为基本费用 15%

(2) 实现思路

抄表员输入抄表日期、设备 id、当前读数和抄表员 id；

系统根据设备 id 找到上次读表时的读数从而计算总费用，生成新的账单。

(3) 核心部分 SQL——找到上次读表时的读数

```
-- 2.1 找到最近抄表记录的id
OPEN read_log_cursor;
find_closest: LOOP
    FETCH read_log_cursor INTO temp_id, temp_time;
    -- SELECT v_finished, temp_id, temp_time;
    IF v_finished = 1 THEN
        LEAVE find_closest;
    END IF;
    IF temp_time > close_time THEN
        SET close_time = temp_time;
        SET close_id = temp_id;
    END IF;
END LOOP find_closest;
CLOSE read_log_cursor;
-- 2.2 根据抄表id找到区间用电量
SELECT read_number INTO begin_num
FROM read_log
WHERE read_id = close_id;

SET degree = r_number - begin_num;
```


(4) 运行结果

```

1  -- 测试语句
2  -- 1.抄表并生成账单
3  CALL read_meter(str_to_date('28-08-2018', '%d-%m-%Y'), 2000, 140, 1);

```

信息	剖析	状态
----	----	----

```

-- 测试语句
-- 1.抄表并生成账单
CALL read_meter(str_to_date('28-08-2018', '%d-%m-%Y'), 2000, 140, 1)
> OK
> 时间: 0.153s

```

3.1.2 用户查询欠费金额

(1) 详细需求

一个用户名下可有多个计费设备，查询欠费清单时，可查询所有计费设备欠费总和，也可查询单个计费设备欠费金额。一个用户在系统内有唯一的 ID，可绑定多张银行卡。

(2) 实现思路

用户输入客户 id，系统会获取当前系统时间，根据计费原则计算该客户所有欠费设备截至当日所应缴纳的费用总和。

(3) 核心部分 SQL——计算滞纳金

```

CREATE DEFINER=`skip-grants user`@`skip-grants host` FUNCTION
`calculate_late`(date DATETIME, paid_fee DECIMAL(9,2), device_type
VARCHAR(2)) RETURNS decimal(9,2)
BEGIN
-- date 账单产生时间
-- paid_fee 账单费用
-- device_type 设备类型
    DECLARE add_price_this_year DECIMAL(9,4); -- 与设备类型关联的附加费 2 比
    DECLARE add_price_other_year DECIMAL(9,4); -- 与设备类型关联的附加费 2 比
    DECLARE temp_fee DECIMAL(9,2);
    DECLARE type_1_this_year DECIMAL(9,4);
    DECLARE type_1_cross_year DECIMAL(9,4);
    DECLARE type_2_this_year DECIMAL(9,4);
    DECLARE type_2_cross_year DECIMAL(9,4);
    -- 获取常量
    SELECT code_value INTO type_1_this_year
    FROM code_table
    WHERE code_key = '居民当年违约金比例';
    SELECT code_value INTO type_1_cross_year
    FROM code_table
    WHERE code_key = '居民跨年违约金比例';
    SELECT code_value INTO type_2_this_year
    FROM code_table
    WHERE code_key = '企业当年违约金比例';
    SELECT code_value INTO type_2_cross_year
    FROM code_table
    WHERE code_key = '企业跨年违约金比例';
    IF device_type='01' THEN
        SET add_price_this_year = type_1_this_year;
        SET add_price_other_year = type_1_cross_year;
    ELSE

```

```

SET add_price_this_year = type_2_this_year;
SET add_price_other_year = type_2_cross_year;
END IF;

IF YEAR(date)=YEAR(NOW()) THEN
SET temp_fee = paid_fee * (TO_DAYS(NOW()) - TO_DAYS(date))
* add_price_this_year;
ELSE
SET temp_fee = paid_fee *
(TO_DAYS(str_to_date((concat('31-12-', YEAR(date))), '%d-%m-%Y')) -
TO_DAYS(date)) * add_price_this_year + paid_fee *
(TO_DAYS(NOW())-TO_DAYS(str_to_date((concat('31-12-', YEAR(date))),
'%d-%m-%Y')) * add_price_other_year;
END IF;
RETURN temp_fee;
END

```

(4) 运行结果

localhost_3306 neugrid 运行 停止 解释

```

1 -- 2.获取用户个人信息
2 SET @c_id = 1000;
3 CALL query_total_fee(@c_id, @total_fee, @c_name, @c_address, @c_balance);
4 SELECT @c_id, @total_fee, @c_name, @c_address, @c_balance;

```

@c_id	@total_fee	@c_name	@c_address	@c_balance
1000	58.89	裴行俭	沈阳海润国际	4.98

3.1.3 缴费

(1) 详细需求

用户缴费，在当月电费清单生成完毕后，用户可进行电费缴纳，缴纳金额可是任意金额。如用户欠费 200 元，用户缴费 250 元，剩余 50 元存入用户余额，如用户缴费 150，则用户再次查询时，欠费金额为 50。

如果用户连续欠费多个月，按照欠费日期依次从缴费金额中扣除，不足一个月的不扣除，余下金额存入用户余额中。

(2) 实现思路

利用游标遍历欠费账单，根据账单“缴费状态”判断应继续缴纳多少钱才能完成缴费。账单有三种缴费状态（未缴费，已缴费和部分缴费）。

(3) 核心部分 SQL——利用游标遍历账单缴费

```

OPEN cost_log_cursor;
pay: LOOP
    FETCH cost_log_cursor INTO temp_cost_id, temp_paid_fee,
temp_actual_fee, temp_late_fee, temp_payable_date, temp_pay_date,
temp_already_fee, temp_pay_state;
    -- SELECT v_finished, new_balance, temp_cost_id,
temp_paid_fee, temp_actual_fee, temp_late_fee, temp_payable_date,
temp_pay_date, temp_already_fee, temp_pay_state;
    IF v_finished = 1 THEN LEAVE pay;
END IF;
    IF new_balance = 0 THEN LEAVE pay;

```

```

        END IF;
-- 如果当前余额足够缴纳当前遍历到的账单
        SET total_fee = calculate_late(temp_payable_date,
temp_paid_fee, temp_device_type) + temp_paid_fee; -- total_fee
-- 当前余额大于待缴费金额
        IF new_balance >= (total_fee - temp_already_fee) THEN
-- 更新客户余额并生成余额变化记录
        SET new_balance = new_balance - (total_fee -
temp_already_fee);
        INSERT INTO balance_log(now_balance, client_id, action,
date) VALUES(new_balance, c_id, '缴费', temp_now); -- 生成新的余额变动记录
        UPDATE client SET balance = new_balance WHERE client_id =
c_id; -- 更新用户余额
-- 更新原有 cost_log
        UPDATE cost_log SET actual_fee = total_fee, late_fee =
total_fee - temp_paid_fee, pay_date = temp_now, already_fee = total_fee,
pay_state = '已缴' WHERE cost_id = temp_cost_id;
-- 增加 change_log
        INSERT change_log(cost_id, actual_fee_1, late_fee_1,
pay_date_1, already_fee_1, pay_state_1, change_time) VALUES(temp_cost_id,
temp_actual_fee, temp_late_fee, temp_pay_date, temp_already_fee,
temp_pay_state, NOW());
        END IF;
-- 当前余额小于待缴费金额, 但不等于 0
        IF new_balance < (total_fee-temp_already_fee) THEN
-- 更新原有 cost_log
        UPDATE cost_log SET pay_date = temp_now, already_fee =
new_balance + already_fee, pay_state = '部分缴' WHERE cost_id = temp_cost_id;
-- 更新客户余额并生成余额变化记录
        SET new_balance = 0; -- 余额一定会被花光变为 0
        INSERT INTO balance_log(now_balance, client_id, action,
date) VALUES(new_balance, c_id, '缴费', temp_now); -- 生成新的余额变动记录
        UPDATE client SET balance = new_balance WHERE client_id =
c_id;
-- 增加 change_log
        INSERT change_log(cost_id, actual_fee_1, late_fee_1,
pay_date_1, already_fee_1, pay_state_1, change_time) VALUES(temp_cost_id,
temp_actual_fee, temp_late_fee, temp_pay_date, temp_already_fee,
temp_pay_state, NOW());
        END IF;
    END LOOP pay;
    CLOSE cost_log_cursor;

```

(4) 运行结果

localhost_3306 neugrid 运行 停止 解释

```

1 -- 3.缴费
2 CALL pay(1002, 1, 'CMB', 2003, @state);
3 SELECT @state;
4
5 CALL pay(1002, 5, 'CMB', 2000, @state);
6 SELECT @state;

```

信息	结果 1	结果 2	结果 3	剖析	状态
@state					
缴费成功					

3.1.4 冲正

(1) 详细需求

用户如果交错了，给其他用户缴费了，在当日 0 点前可以冲正，即把钱收回，放入余额，过了 0 点缴费生效，不能冲正。

冲正交易为缴费交易的逆向操作，即取消缴费操作，取消前，判断是否能找到原银行流水，客户号，实收金额的收费记录，没找到，返回冲正错误的提示信息。如果找到，取消缴费。

(2) 实现思路

用户输入流水 id，系统根据这一 id 找到对应的账单 id 及信息变化记录。冲正时，系统先根据余额变化记录将费用缴回至用户的余额中，然后根据信息变化记录将账单信息恢复到误付款前的状态。

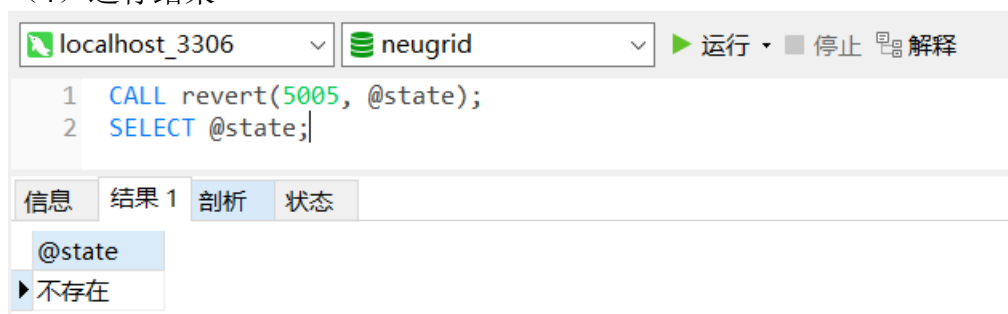
(3) 核心 SQL——给原客户账户返钱

```
-- 2. 给原客户账户返钱
SELECT client_id, pay_amount, bank_id, client_id INTO temp_client_id,
temp_pay_amount, temp_bank_id, temp_device_id
FROM pay_log
WHERE transfer_id = t_id AND TO_DAYS(pay_time)=TO_DAYS(NOW()) AND
pay_amount > 0;

SELECT balance INTO temp_balance
FROM client
WHERE client_id = temp_client_id;
-- 2.1 查找所付款
SELECT pay_time INTO temp_pay_time
FROM pay_log
WHERE transfer_id = t_id AND pay_amount > 0;
-- 根据余额变化表计算花的钱
SELECT MAX(now_balance) - MIN(now_balance) INTO real_pay_amount
FROM balance_log
WHERE date = temp_pay_time;

UPDATE client SET balance = temp_balance + real_pay_amount WHERE client_id
= temp_client_id;
INSERT balance_log(now_balance, client_id, action, date)
VALUES(temp_balance + real_pay_amount, temp_client_id, '冲正', NOW());
```

(4) 运行结果



localhost_3306 neugrid 运行 停止 解释

```
1 CALL revert(5005, @state);
2 SELECT @state;
```

信息 结果 1 剖析 状态

@state
不存在

3.1.5 对账

(1) 详细需求

a. 银行对账（对总帐）

每天凌晨银行会发送所有前一天的总交易金额，总交易笔数，及所有交易金额等明细信息，供企业方核对。银行传来的信息包括：银行代码，总笔数，总金额，对帐日期。

根据银行代码，对帐日期在相关表中查找所有满足当前银行，日期的缴费记录笔数和金额，同银行传来数据做比较，如果总笔数，总金额都相同，可以认为此次对总账成功，否则调用对明细账模块。

b. 银行对账（对明细）

总账不平需要逐笔核对明细账。判断账务不平的原因，并记录到对账异常表中，由其他应用程序来处理。

根据银行传来的银行代码，银行交易流水号，交易日期时间，客户号，交费金额等信息，在业务系统表中查找相应的记录，对账不平的可能性有企业方无此流水号的缴费信息，银行无此流水号的缴费信息，金额不等，把所有对账不平的信息及原因记录到对账异常表。

(2) 实现思路

用户输入欲对账的时间，系统利用 left join/union 等 sql 操作代替 MySQL 中没有的 full join；

(3) 核心 SQL 代码——生成待对账表

```

DECLARE error_log_cursor CURSOR FOR
SELECT A.transfer_id, B.transfer_id, A.pay_amount, B.transfer_amount,
A.pay_time, B.transfer_time, A.client_id, B.client_id
FROM transfer_log B LEFT JOIN (SELECT * FROM pay_log WHERE pay_type = '缴费') A ON A.transfer_id = B.transfer_id
WHERE B.bank_id = b_id AND TO_DAYS(B.transfer_time) = TO_DAYS(date);

DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished = 1;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SELECT 'Exception';

-- 2. 生成错误记录
OPEN error_log_cursor;
find_error: LOOP
    FETCH error_log_cursor INTO temp_A_transfer_id, temp_B_transfer_id,
temp_A_pay_amount, temp_B_transfer_amount, temp_A_pay_time,
temp_B_transfer_time, temp_A_client_id, temp_B_client_id;
    IF v_finished = 1 THEN LEAVE find_error;
    END IF;

    IF ISNULL(temp_A_transfer_id) = 1 THEN INSERT
error_log(account_time, bank_id, transfer_id, client_id, bank_amount,
enterprise_amount, error_type) VALUES (temp_B_transfer_time, b_id,
temp_B_transfer_id, temp_B_client_id, temp_B_transfer_amount, 0, '企业方无流水信息');
    ELSEIF ISNULL(temp_B_transfer_id) = 1 THEN INSERT
error_log(account_time, bank_id, transfer_id, client_id, bank_amount,
enterprise_amount, account_info) VALUES (temp_A_pay_time, b_id,
temp_A_transfer_id, temp_A_client_id, 0, temp_A_pay_amount, '银行方无流水信息');
    ELSEIF temp_A_pay_amount <>
temp_B_transfer_amount THEN INSERT error_log(account_time, bank_id,
transfer_id, client_id, bank_amount, enterprise_amount, account_info)

```

```
VALUES (temp_A_pay_time, b_id, temp_A_transfer_id,
temp_A_client_id, temp_B_transfer_amount, temp_A_pay_amount, '金额不等');
END IF;
END LOOP find_error;
CLOSE error_log_cursor;
```

(4) 运行结果

localhost_3306 neugrid 运行 停止 解释

```
1 -- 5.对总账
2 CALL check_total('CMB', 3, 300, str_to_date('20-08-2018', '%d-%m-%Y'), @state);
3 SELECT @state;
4
5 CALL check_total('CMB', 2, 250, str_to_date('20-08-2018', '%d-%m-%Y'), @state);
6 SELECT @state;
7
8 -- 6.对明细
9 CALL check_detail('CMB', str_to_date('20-08-2018', '%d-%m-%Y'));
```

信息	结果 1	结果 2	剖析	状态
@state				
error				

3.2 特殊查询功能实现

3.2.1 查询出所有欠费用户

```
SELECT DISTINCT device.client_id, client_name
FROM cost_log JOIN (device JOIN client USING(client_id)) USING(device_id)
WHERE pay_state = '未缴' OR pay_state = '部分缴';
/* Runnig Result
client_id      client_name
1000          裴行俭
1002          潘育龙
1004          周亚夫
*/
```

3.2.2 查询出拥有超过 2 个设备的用户

```
SELECT client_id, client_name, address, balance
FROM device JOIN client USING(client_id)
GROUP BY client_id
HAVING COUNT(*) > 2;
/* Runnig Result
client_id      client_name      address balance
1002          潘育龙          沈阳长白岛      5.00
*/
```

3.2.3 统计电力企业某天的总应收费用，实收费用

```
-- Q3: 统计电力企业某天的总应收费用，实收费用
SELECT (SELECT sum(calculate_late(payable_date, paid_fee, device_type) +
paid_fee - already_fee)
FROM cost_log JOIN device USING(device_id)
```

```

WHERE pay_state <> '已缴' AND
TO_DAYS(payable_date) < TO_DAYS(str_to_date('20-02-2019', '%d-%m-%Y')) AS
'总应收费用',
      (SELECT sum(pay_amount)
       FROM pay_log
       WHERE TO_DAYS(pay_time) =
TO_DAYS(str_to_date('20-02-2019', '%d-%m-%Y')) AND pay_type = '缴费') AS '
总实收费用';
/* Running Result
总营收费用 总实收费用
296.26 500
*/

```

3.2.4 查询出所有欠费超过半年的用户

```

SELECT client_id, client_name
FROM cost_log JOIN (device JOIN client USING(client_id)) USING(device_id)
WHERE (pay_state = '未缴' OR pay_state = '部分缴') AND TIMESTAMPDIFF(MONTH,
payable_date, NOW()) >= 6;
/* Runnig Result
client_id      client_name
1000      裴行俭
*/

```

3.2.5 查询任意用户的欠费总额

```

SELECT client_id, client_name, sum(paid_fee) AS '欠费总额', sum(paid_fee +
calculate_late(payable_date, paid_fee, device_type)) AS '欠费总额(含滞纳金)'
FROM cost_log JOIN (device JOIN client USING(client_id)) USING(device_id)
WHERE (pay_state = '未缴' OR pay_state = '部分缴')
GROUP BY client_id;
-- AND client_id = '1000';
/* Runnig Result
client_id      client_name      欠费总额 欠费总额(含滞纳金)
1000      裴行俭      52.04      58.78
1002      潘育龙      101.19      105.47
1004      周亚夫      56.66      58.30
*/

```

3.2.6 查询出某个月用电量最高的3名用户

```

SELECT MONTH(A.date) AS 'MONTH', client_id, client_name
FROM cost_log A JOIN client USING(client_id)
WHERE (SELECT count(client_id)
      FROM cost_log B
      WHERE (MONTH(A.date) = MONTH(B.date)) AND B.end_number -
B.begin_number > A.end_number - A.begin_number) <= 2;
/* Running Result
client_id      client_name      用电量
1002      潘育龙      150
1005      用户1      100
1001      秦良玉      40
*/

```

3.2.7 查询出电力企业某个月哪天的缴费人数最多

```

SELECT YEAR(pay_time), MONTH(pay_time), DAY(pay_time), count(DAY(pay_time))
AS '缴费人数'
FROM pay_log
WHERE MONTH(pay_time) = 8 AND YEAR(pay_time) = 2018
GROUP BY (DAY(pay_time))
ORDER BY count(DAY(pay_time))
LIMIT 0,1;

/* Runnig Result
YEAR(pay_time)  MONTH(pay_time) DAY(pay_time)  缴费人数
2018      8      20      3
*/

```

3.2.8 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

```

SELECT device_type, count(*) AS 'number'
FROM device JOIN client USING(client_id)
GROUP BY device_type
ORDER BY number DESC;

/* Runnig Result
device_type      number
01              5
02              3
*/

```

3.2.9 统计每个月各银行缴费人次，从高到低排序

```

SELECT MONTH(transfer_time) AS 'month', bank_id, COUNT(*) AS 'number'
FROM transfer_log
GROUP BY MONTH(transfer_time), bank_id
ORDER BY number DESC

/* Runnig Result
month  bank_id number
8      CMB      2
8      ICBC     1
*/

```

3.2.10 查询出电力企业所有新增用户（使用设备不足半年）

```

SELECT client_id, client_name
FROM read_log JOIN (client JOIN device USING(client_id)) USING(device_id)
GROUP BY client_id
HAVING MAX(MONTH(NOW()) - MONTH(read_date)) <= 6;

/* Running Result
client_id      client_name
1001          秦良玉
1002          潘育龙
1004          周亚夫
*/

```


4. 程序实现

4.1 技术框架

	解决方案	
前端	动态展示数据	利用 Vue.js 技术实现数据绑定、从后端请求数据等操作
	界面样式	利用 Bootstrap 框架设计好看的用户界面和良好的交互体验
后端	程序语言	Java
	数据库	MySQL5.7
	数据交互	JSON
	返回数据给前端	利用最新的 Springboot 开发框架，实现前后端开发的完全分离

4.2 JDBC 连接数据库

4.2.1 良好的编程风格

数据库使用时会不断的开启与断开和数据库的连接，为了减少冗余的代码，我预先编写了 connect()和 disconnect()函数，同时还有 before()和 after()函数用于完成数据库查询语句的初始化。良好的编程风格使代码整体结构更清晰。

```

/* 基础操作 */
// 连接数据库
private static void connect() throws ClassNotFoundException,
SQLException {
    if (conn == null || conn.isClosed()) { // 若连接为空或已关闭
        Class.forName(JDBC_DRIVER);
        System.out.println("[INFO] 正在尝试连接数据库 | Trying to
connect with the database...");
        conn = DriverManager.getConnection(DB_URL, USER, PASS);
        if (!conn.isClosed()) {
            prepareSql();
        } else {
            System.out.println("[WARNING] 数据库连接失败 | Database
Connection Failed");
        }
    }
}

// 断开数据库
private static void disconnect() throws SQLException {
    if (!conn.isClosed()) {
        conn.close();
        System.out.println("[INFO] 数据库连接已断开 | Database
disconnectes successfully");
    }
}

// 前置操作
private static void before() throws SQLException, ClassNotFoundException
{
    connect();
    stmt = conn.createStatement();
}

```

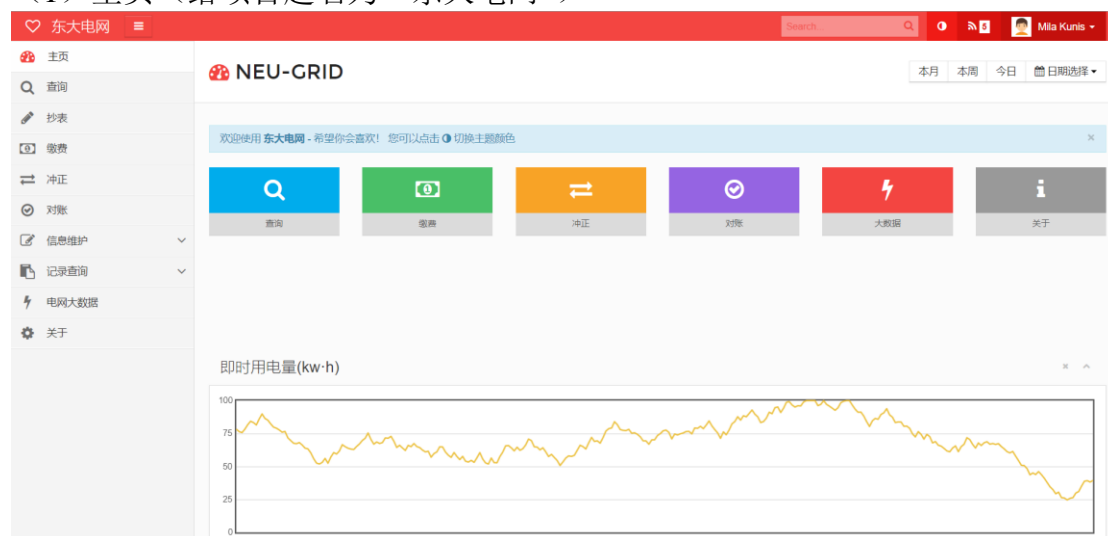
```
}  
// 后置操作  
private static void after() throws SQLException {  
    // disconnect(); // 暂时保持长连接  
}
```

4.2.2 防止 SQL 注入的安全考量

与此同时，因为我开发的系统是基于 web 的，必然会遇到 SQL 注入的潜在安全问题，所以我将全部 sql 语句预编译，利用 Java PreparedStatement 类所提供的 prepareStatement()方法设置参数。

4.3 界面展示

(1) 主页（给项目起名为“东大电网”）



(2) 查询客户信息

a. 展示全部客户

东大电网

客户搜索

全部客户

10 records per page

序号	ID	姓名	地址	余额	操作
1	1000	裴行俭	沈阳海润国际	4.98	确定
2	1001	秦良玉	沈阳碧桂园银河城	130.32	确定
3	1002	潘育龙	沈阳长白岛	101.45	确定
4	1003	卫青	沈阳华强城	90.81	确定
5	1004	周亚夫	沈阳恒大绿洲	80.00	确定

Showing 1 to 5 of 5 entries

b. 点击右侧按钮选择用户，下方展示用户信息

51004周亚夫沈阳恒大绿洲80.00确定

序号ID姓名地址

Showing 1 to 5 of 5 entriesPrevious1Next

潘育龙ID: 1002

余额: ¥101.45住址: 沈阳长白岛

Code	Item	账单ID	设备ID	日期	基础花	附加费1	附加费2	待支付	实际支付	违约金	可付款日期	付款日期	已付费用	付款状态
4000	2006	2018-07-07 00:00:00.0	0	64	¥31.36	¥2.51	¥4.70	¥38.57	¥40.19	¥1.62	2018-07-31 00:00:00.0	2018-08-19 19:54:56.0	40.19	已缴
4002	2003	2018-06-05 00:00:00.0	0	75	¥36.75	¥2.94	¥3.68	¥43.37	¥	¥	2018-06-30 00:00:00.0		0.00	未缴
4003	2003	2018-07-05 00:00:00.0	75	175	¥49.00	¥3.92	¥4.90	¥57.82	¥	¥	2018-07-31 00:00:00.0		0.00	未缴

欠费总额: ¥139.76

应收费用=基本费用+附加费用1+附加费用2

(3) 抄表

东大电网

抄表

抄表日期08/20/2018 03:12:24 AM

设备ID2000

当前读数140

抄表员ID1

提交

August 2018

Su	Mo	Tu	We	Th	Fr	Sa
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

提交

(4) 缴费

缴费

客户ID1000

缴费金额¥100.00

选择银行

中国建设银行China Construction Bank

招商银行CHINA MERCHANTS BANK

中国工商银行INDUSTRIAL AND COMMERCIAL BANK OF CHINA

设备ID2000

提交

(5) 冲正

东大电网

冲正

流水ID

5003

提交

Lorem ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec nisl est, vulputate at porttitor non, interdum a mauris. Phasellus imperdiet gravida pulvinar.

(6) 对账

a. 核对总账

核对总账

选择银行

中国建设银行
China Construction Bank

招商银行
CHINA MERCHANTS BANK

中国工商银行
INDUSTRIAL AND COMMERCIAL BANK OF CHINA

交易笔数

3

交易总金额

¥ 500.00

对账日期

read_date_1

提交

August 2018

Su Mo Tu We Th Fr Sa

29 30 31 1 2 3 4

20

21 22 23 24 25

26 27 28 29 30 31 1

08/20/2018 03:17:53 AM

提交

Lorem ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec nisl est, vulputate at porttitor non, interdum a mauris. Phasellus imperdiet gravida pulvinar.

b. 核对明细

核对明细

选择银行

中国建设银行
China Construction Bank

招商银行
CHINA MERCHANTS BANK

中国工商银行
INDUSTRIAL AND COMMERCIAL BANK OF CHINA

交易笔数

3

交易总金额

¥ 500.00

对账日期

08/20/2018 03:17:53 AM

提交

August 2018

Su Mo Tu We Th Fr Sa

29 30 31 1 2 3 4

5 6 7 8 9 10 11

12 13 14 15 16 17 18

19 20 21 22 23 24 25

26 27 28 29 30 31 1

08/20/2018 03:17:53 AM

提交

Lorem ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec nisl est, vulputate at porttitor non, interdum a mauris. Phasellus imperdiet gravida pulvinar.

(7) 信息维护（附加功能）

a. 查看全部信息（以客户信息管理为例）

客户信息管理

全部客户

10 records per page

Search:

序号	ID	姓名	地址	余额	操作
1	1000	裴行俭	沈阳海润国际	4.98	✓ ✗
2	1001	秦良玉	沈阳碧桂园银河城	130.32	✓ ✗
3	1002	潘育龙	沈阳长白岛	101.45	✓ ✗
4	1003	卫青	沈阳华强城	90.81	✓ ✗
5	1004	周亚夫	沈阳恒大绿洲	80.00	✓ ✗

Showing 1 to 5 of 5 entries

Previous 1 Next

添加客户

b. 增加新的记录

添加客户

ID: 1005

姓名: 许瑞文

地址: 上海汤臣一品

初始余额: 0

添加

(8) 关于（附加功能）

关于

开源地址

Logo

项目背景

项目来源于 东北大学软件学院2018年数据库应用程序设计实践课程

项目说明

项目包含服务器端，网页客户端和微信小程序客户端。

- 网页客户端提供管理功能；
- 微信小程序提供用户缴费和抄表员抄表功能，满足他们对移动使用的需求。

5. 遇到的问题及其解决方案

5.1 数据库方面

5.1.1 因为发生错误而停止循环

(1) 问题

当我在使用游标对结果集遍历时，发现只运行了一次程序就不再运行。

(2) 解决方案

我猜测是遇到了异常，于是参考文档上的建议，在存储过程开始处声明一个错误处理程序，以便能让程序暂时忽略这一错误，继续运行。

错误处理声明如下：

```
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SELECT 'Exception';
```

5.1.2 没有 debug 功能该如何寻找错误

(1) 问题

当我在开发测试存储过程时，发现 MySQL 没有 debug 功能，这导致有些问题很难定位。

(2) 解决方案

在有可能出现问题的代码前后增加 SELECT 语句，类似于我们在开发 Java 程序中经常使用的 System.out.println(); 观察输出结果可以判断程序在哪里停止。

5.2 JDBC 及界面开发方面

5.2.1 string 与 date 的转换

(1) 问题

我从界面返回来的数据类型是 string，但是存入数据库时必须是 date 类型，利用 java.util 包内的 date 转换工具并不能满足要求；

(2) 解决方案

引入了 sql.date 的 maven 包，利用 timestamp 对象可以实现上述转换。

```
CallableStatement cstmt = conn.prepareCall("{CALL read_meter(?, ?, ?, ?)}");  
cstmt.setTimestamp(1, new Timestamp(Long.valueOf(read_date)));
```

6. 创新点

6.1 用户界面

本系统的用户界面是系统的一大创新。整个系统在完成数据库实验要求内容的基础上，利用 Springboot 框架搭建了后端，利用 Vue.js 从前端请求后端数据并进行列表和条件渲染，利用 Bootstrap UI 框架美化前端页面。最终的展现形式是一个 Web 应用，网页布局合理，界面美观，用户体验良好。

6.2 系统兼具两套交互模式

本系统兼具命令行和 Web 应用两套交互模式。通过命令行去测试底层数据，通过 Web 应用使得系统最终有一个良好的展现效果。利用 mode 标记区分这两套交互模式，使其不冲突，完美并行。同时，当运行 Web 应用时，命令行还会展示一些记录数据，方便开发。

下图为系统运行时的运行记录。我做了大量的提示性输出，便于后续的调试。

```
2018-08-31 03:07:59.920 INFO 6668 --- [main] o.s.j.e.a.AnnotationMBeanExporter : Registering beans for
2018-08-31 03:08:02.708 INFO 6668 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port
2018-08-31 03:08:02.728 INFO 6668 --- [main] cn.edu.neu.NeuGridBackendApplication : Started NeuGridBacker
2018-08-31 03:10:05.535 INFO 6668 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring F
2018-08-31 03:10:05.536 INFO 6668 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : FrameworkServlet 'dis
2018-08-31 03:10:05.556 INFO 6668 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : FrameworkServlet 'dis
[INFO] /api/getAllClients 收到访问请求, 正在处理
[INFO] 正在尝试连接数据库 | Trying to connect with the database...
Fri Aug 31 03:10:29 CST 2018 WARN: Establishing SSL connection without server's identity verification is not recommended.
[INFO] /api/getAllClients 收到访问请求, 正在处理
[INFO] /api/getTotalPaidFeeByClient_ID 收到访问请求, 正在处理
[INFO] /api/getCostsByClient_ID 收到访问请求, 正在处理
[INFO] /api/getClientByClientID 收到访问请求, 正在处理
com.mysql.jdbc.JDBC42PreparedStatement@3fecccc: SELECT * FROM cost_log WHERE DEVICE_ID IN (SELECT DEVICE_ID FROM device
```

6.3 系统具有较强的鲁棒性

系统内部具有完备的错误检测机制, 例如用户缴费时输入负数, 系统会提示用户“缴费金额必须大于0”; 再例如用户错误地输入了不存在的设备ID, 系统也会将错误信息及时反馈给用户。

6.4 数据库设计工具 powerdesign 的学习使用

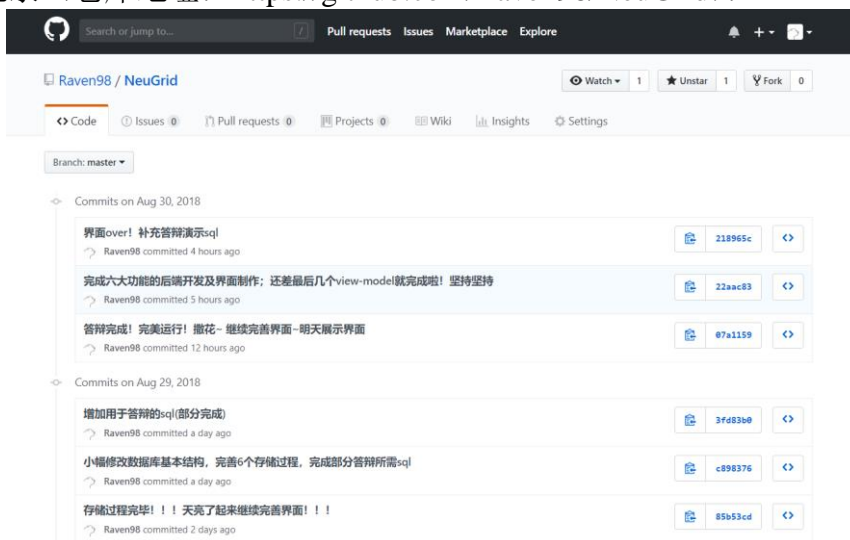
本次实验课在数据库设计阶段我使用了 powerdesigner 来进行设计, 通过最专业的数据库设计软件我不断分析我的逻辑模式的优劣不断完善。

6.5 安全开发, 防止 SQL 注入

因为我开发的系统是基于 web 的, 必然会遇到 SQL 注入的潜在安全问题, 所以我将全部 sql 语句预编译, 利用 Java PreparedStatement 类所提供的 prepareStatement()方法设置参数。

6.6 版本控制工具的灵活运用

本次实验课期间我一直在用 Git 将我的本地项目仓库同步上传至远程的 Github 仓库。实验课期间共有 22 次提交记录。通过使用版本控制工具, 我能很好地把握整个项目进度, 同时每次提交也是对目前项目进展的一个总结与反思, 帮助我有条不紊地完成数据库的设计、实现到最后的完整前后端交互。下图为历史提交记录 (仓库地址: <https://github.com/Raven98/NeuGrid>):



7. 总结

又到了每学期最喜欢的实验课啦。每次实验课都感觉自己的技术能力都会有很大提升，这次也不例外。

这次实验课的主题是数据库。在上这次实验课前我也有一段使用数据库的经历了，不过还从来没使用过存储过程，每次都是直接在 Java 代码里写 sql 语句配合着 Java 代码实现需求。现在看来这种方式不仅仅繁琐复杂更面临这 SQL 注入等安全威胁。这次实验课学到最多的就是数据库的存储过程开发方面，以后也要多使用存储过程，提高开发效率。

这次实验我还花费了很多时间用来制作用户界面。在开始制作界面前，我和同学们认真交流了一下，最终选定利用全新的 Springboot 框架搭配非常流行的 Vue.js 框架完成 Web 应用的开发。选择这样的技术组合主要是因为他们可以实现前后端开发的完全分离，这样我之前已经完成了的数据库存储过程的代码就可以很好地被重用。

另外，这次实验课正恰逢 2018 级的学弟学妹们入学。转瞬之间发现自己已经大三毕业了。站在大三这个时间点，感到对自己未来的职业发展方向很迷茫。通过前两年的学习，我学到了大量计算机和软件开发的基本知识。大三开始后我们所学的内容可能会更专业更深入，面对新的挑战我必须要做好准备，永远保持热爱学习的心态，努力去找找到自己真正热爱的计算机方向并深入下去。

最后，谢谢各位老师的认真指导，我定将珍惜每次实验课的宝贵机会，认真完成每次实验课的任务，努力提升自己，在计算机的世界寻找更多乐趣！

参考文献

- [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			