中山大学数据科学与计算机学院本科生实验报告

(2019 年秋季学期)

课程名称：区块链原理与技术　　任课教师：郑子彬

年级	17 级	专业（方向）	软件工程
学号	17343087	姓名	孟印真
电话		Email	
开始日期	2019/11/15	完成日期	2019/12/9

一、 项目背景

基于已有的开源区块链系统 FISCO-BCOS（https://github.com/FISCO-BCOS/FISCO-BCOS），以联盟链为主，开发基于区块链或区块链智能合约的供应链金融平台，实现供应链应收账款资产的溯源、流转。

1. 区块链+供应链金融：

将供应链上的每一笔交易和应收账款单据上链，同时引入第三方可信机构来确认这些信
息的交易，例如银行，物流公司等，确保交易和单据的真实性。同时，支持应收账款的转让，
融资，清算等，让核心企业的信用可以传递到供应链的下游企业，减小中小企业的融资难度。

2. 实现功能：

功能一：实现采购商品—签发应收账款 交易上链。例如车企从轮胎公司购买一批轮胎并
签订应收账款单据。

功能二：实现应收账款的转让上链，轮胎公司从轮毂公司购买一笔轮毂，便将于车企的
应收账款单据部分转让给轮毂公司。轮毂公司可以利用这个新的单据去融资或者要求车企到

期时归还钱款。

功能三：利用应收账款向银行融资上链，供应链上所有可以利用应收账款单据向银行申

请融资。

功能四：应收账款支付结算上链，应收账款单据到期时核心企业向下游企业支付相应的欠

款。

3. 加分项

a. 功能：除了必要功能之外，实现更多的功能，限定：供应链金融领域相关

b. 底层：改进区块链平台底层或自行开发区块链（共识算法等）

c. 合约：实现链上数据隐私保护（同态加密、属性加密等）

d. 前端：友好高效的用户界面

e. 其他有挑战性的创新

二、 方案设计

存储设计、数据流图、核心功能介绍（文字+代码）

存储设计：使用的是 api 中给的 table 来存储用户信息，以及交易信息。

用户信息

```
tf.createTable("t_asset", "account", "asset_value");
```

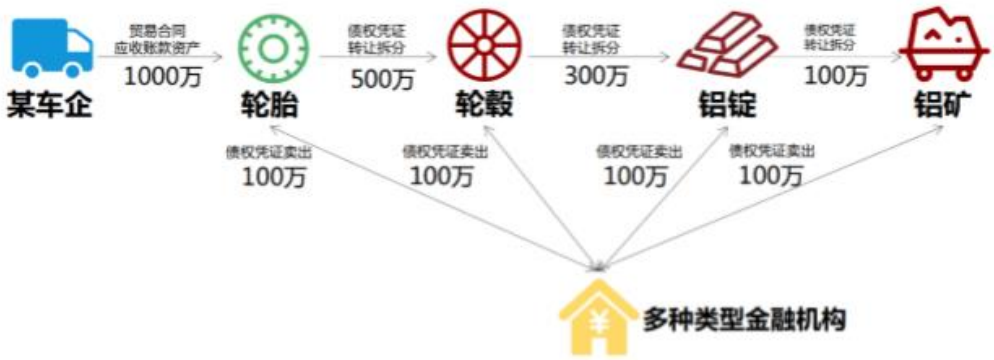
account 为用户姓名，asset_value 为用户余额

交易信息：

```
tf.createTable("t_receipt", "num", "account_from,acount_to, value, start_time, range");
```

account_from 转出用户，acount_to 转入用户，value 转账金额，start_time 交易的时间，range（生还款时间），num 交易编号。

数据流图:



核心功能介绍：

注册：

```
function register(string account, uint256 asset_value) public returns(int256){
    int256 ret_code = 0;
    int256 ret= 0;
    uint256 temp_asset_value = 0;
    // 查询账户是否存在
    (ret, temp_asset_value) = select(account);
    if(ret != 0) {
        Table table = openTable();

        Entry entry = table.newEntry();
        entry.set("account", account);
        entry.set("asset_value", int256(asset_value));
        // 插入
        int count = table.insert(account, entry);
        if (count == 1) {
            // 成功
            ret_code = 0;
        } else {
            // 失败? 无权限或者其他错误
            ret_code = -2;
        }
    } else {
        // 账户已存在
        ret_code = -1;
    }

    emit RegisterEvent(ret_code, account, asset_value);

    return ret_code;
}
```

查询：

```
    function select(string account) public constant returns(int256, uint256) {
        // 打开表
        Table table = openTable();
        // 查询
        Entries entries = table.select(account, table.newCondition());
        uint256 asset_value = 0;
        if (0 == uint256(entries.size())) {
            return (-1, asset_value);
        } else {
            Entry entry = entries.get(0);
            return (0, uint256(entry.getInt("asset_value")));
        }
    }
```

转账：

```
function transfer(string from_account, string to_account, uint256 amount) public returns(int256) {
    // 查询转移资产账户信息
    int ret_code = 0;
    int256 ret = 0;
    uint256 from_asset_value = 0;
    uint256 to_asset_value = 0;

    // 转移账户是否存在?
    (ret, from_asset_value) = select(from_account);
    if(ret != 0) {
        ret_code = -1;
        // 转移账户不存在
        emit TransferEvent(ret_code, from_account, to_account, amount);
        return ret_code;

    }

    // 接受账户是否存在?
    (ret, to_asset_value) = select(to_account);
    if(ret != 0) {
        ret_code = -2;
        // 接收资产的账户不存在
        emit TransferEvent(ret_code, from_account, to_account, amount);
        return ret_code;
    }

    if(from_asset_value < amount) {
        ret_code = -3;
        // 转移资产的账户金额不足
        emit TransferEvent(ret_code, from_account, to_account, amount);
        return ret_code;
    }

    if (to_asset_value + amount < to_asset_value) {
        ret_code = -4;
        // 接收账户金额溢出
        emit TransferEvent(ret_code, from_account, to_account, amount);
        return ret_code;
    }

    Table table = openTable();

    Entry entry0 = table.newEntry();
    entry0.set("account", from_account);
    entry0.set("asset_value", int256(from_asset_value - amount));
    // 更新转账账户
    int count = table.update(from_account, entry0, table.newCondition());
    if(count != 1) {
        ret_code = -5;
        // 失败? 无权限或者其他错误?
        emit TransferEvent(ret_code, from_account, to_account, amount);
        return ret_code;
    }

    Entry entry1 = table.newEntry();
    entry1.set("account", to_account);
    entry1.set("asset_value", int256(to_asset_value + amount));
    // 更新接收账户
    table.update(to_account, entry1, table.newCondition());

    emit TransferEvent(ret_code, from_account, to_account, amount);

    return ret_code;
}
```

三、 功能测试

Register:

结果：

{"transactionHash":"0xf69ac1b4253153ca5d7ede687c54e24936ed8ec2a892c6f4a0f43b8785141fee","status":"0x0","output":{"0":"0"}}

查询：

{"status":"0x0","output":{"0":"0","1":"2710"}}

此处 2710 为 16 进制表示。可以看出

转账：

查询 myz 的余额

{"status":"0x0","output":{"0":"0","1":"2648"}}

进行转账：

{"transactionHash":"0xdca390727f044f21d1b96a7b75d70f1fdfac74bea6f74be978835bfa35433380","status":"0x0","output":{"0":"0"}}

再次查询两个用户的余额：

myz：

{"status":"0x0","output":{"0":"0","1":"2a30"}}

baidu:

{"status":"0x0","output":{"0":"0","1":"2328"}}

可以看出转账成功

四、 界面展示

五、 心得体会

总的来说，感觉特别难，首先是对基本概念的不太理解，在刚开始进行时都是跟着教程来做，没有深入了解其内容的含义，导致在做第三部分时几乎无从下手，依靠博客和同学指导才有一点方向。

通过这次作业，我了解了基于区块链的金融体系的优点，学会了智能合约的编写以及部署，也了解了智能合约的好处，也对前后端有了进一步的了解，大大提高了自己的代码能力。