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	16	专业（方向）	软件工程数媒
学号	16340065	姓名	谷田
电话	15989003409	Email	1753664673@qq.com
开始日期	2019-12-15	完成日期	2020-01-17

项目背景

传统供应链金融：

某车企（宝马）因为其造车技术特别牛，消费者口碑好，所以其在同行业中占据绝对优势地位。因此，在金融机构（银行）对该车企的信用评级将很高，认为他有很大的风险承担的能力。在某次交易中，该车企从轮胎公司购买了一批轮胎，但由于资金暂时短缺向轮胎公司签订了 1000 万的应收账款单据，承诺 1 年后归还轮胎公司 1000 万。这个过程可以拉上金融机构例如银行来对这笔交易作见证，确认这笔交易的真实性。在接下里的几个月里，轮胎公司因为资金短缺需要融资，这个时候它可以凭借跟某车企签订的应收账款单据向金融结构借款，金融机构认可该车企（核心企业）的还款能力，因此愿意借款给轮胎公司。但是，样的信任关系并不会往下游传递。在某个交易中，轮胎公司从轮毂公司购买了一批轮毂，但由于租金暂时短缺向轮胎公司签订了 500 万的应收账款单据，承诺 1 年后归还轮胎公司 500 万。当轮毂公司想利用这个应收账款单据向金融机构借款融资的时候，金融机构因为不认轮胎公司的还款能力，需要对轮胎公司进行详细的信用分析以评估其还款能

力同时验证应收账款单据的真实性，才能决定是否借款给轮毂公司。这个过程将增加很多经济成本，而这个问题主要是由于该车企的信用无法在整个供应链中传递以及交易信息不透明化所导致的。

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。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。

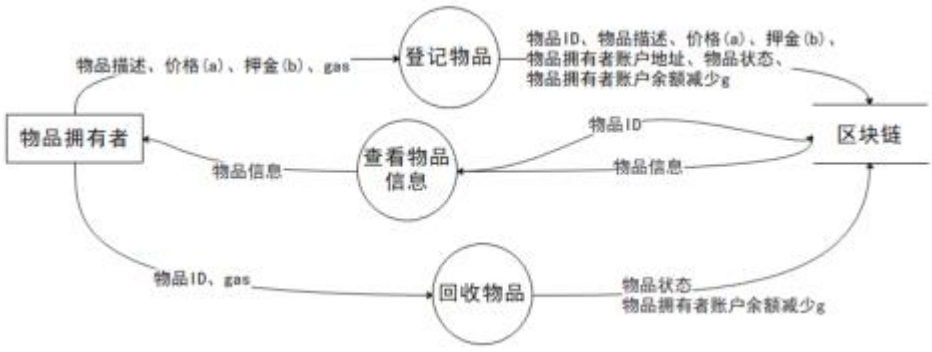
功能三：利用应收账款向银行融资上链，供应链上所有可以利用应收账款单据向银行申请融资。

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。

方案设计

存储设计、数据流图、核心功能介绍（文字+代码）

数据流图示例：



```
    string myName;
    int myAmount;
    string key;

    constructor () {
        myName = "Bank";
        myAmount = int(1000000);
        key = "receipt";
        createTable();
        insertCompany(myName, int(0));
    }
    // Company
    struct Company {
        string name;
        int money;
        int companyType; // 0 is bank
    }
    Company company;
    // Receipt
    struct Receipt {
        int key;
        string debtorName;
        string debteeName;
        int amount;
        int deadline;
    }
    Receipt receipt;
```

一个公司实体：name 即公司名称， money 为资金，companyType 记为公司类型，0 代表央行。1 代表普通公司。Receipt 是一个交易，或者一个账单，key 为它的关键码，debtorName 与 debteeName 分别表示债务人和债权人，amount 是债务数额，deadline 即到期日期。

```
function createTable() private {
    TableFactory tf = TableFactory(0x1001);
    tf.createTable("t_company", "name", "companyType,money");
    tf.createTable("t_receipt", "key", "debtorName,debteeName,amount,deadline");
}

function openTable(string tableName) private returns(Table) {
    TableFactory tf = TableFactory(0x1001);
    Table table = tf.openTable(tableName);
    return table;
}
```

创建表格与获取表格，

```
// event about Company Register
event CompanyRegister (string name);
event findCompany (string name, int companyType);
event createReceipt (string debtor, string debtee, int amount, int deadline);
event findReceipt (string debtor, string debtee, int amount, int deadline);
event removeR(int count);
event UpdateR(int count);
event IsDeadline(string debtor, string debtee, int amount);
```

各项事件 event

```
function isStringEQ (string a, string b) private returns(bool) {
    if (bytes(a).length != bytes(b).length) {
        return false;
    }
    for(uint i = 0;i < bytes(b).length;i++) {
        if(bytes(a)[i] != bytes(b)[i]) {
            return false;
        }
    }
    return true;
}
```

判断字符串是否相等

```
// Company Register
function registerCompany(string name) public returns(string, int){
    Table t_company = openTable("t_company");
    Condition condition = t_company.newCondition();
    Entries entries = t_company.select(name, condition);
    require(entries.size() == 0, "Company should exist and be unique");
    insertCompany(name, int(1));
    emit CompanyRegister(name);
    return (name, int(1));
}

function selectCompany(string name) public returns(string, int){
    TableFactory tf = TableFactory(0x1001);
    Table table = tf.openTable("t_company");

    Condition condition = table.newCondition();
    Entries entries = table.select(name, condition);
    require(entries.size() == 1, "Company should exist and be unique");
    Entry entry = entries.get(0);
    company.companyType = entry.getInt("companyType");
    company.name = entry.getString("name");
    //company.money = entry.getInt("money");
    emit findCompany(company.name, company.companyType);
    return (name, int(1));
}

function insertCompany (string name, int companyType) private {
    TableFactory tf = TableFactory(0x1001);
    Table t_company = tf.openTable("t_company");
    //Entries entries = t_company.select(toString(addr), t_company.newCondition());
    //require(entries.size() == 0, "Company should not exist");
    Entry entry = t_company.newEntry();
    entry.set("name", name);
    entry.set("companyType", companyType);
    //entry.set("money", int(10000));
    t_company.insert(name, entry);
}
```

通过公司名注册公司和搜索公司

```
function insertReceipt(string debtor, string debtee, int amount, int deadline) public retu
    Table t_receipt = openTable("t_receipt");
    Condition condition1 = t_receipt.newCondition();
    condition1.EQ("debteeName", debtee);
    condition1.EQ("debtorName", debtor);
    Entries entries = t_receipt.select(key, condition1);
    require(entries.size() == 0, "Debtor haven't paid the money debtor owe!");
    Entry entry = t_receipt.newEntry();
    entry.set("debtorName", debtor);
    entry.set("debteeName", debtee);
    entry.set("amount", amount);
    entry.set("deadline", deadline);
    t_receipt.insert(key, entry);
    emit createReceipt(debtor, debtee, amount, deadline);
    return (debtor, debtee, amount, deadline);
}


function removeReceipt(string debtor, string debtee) public returns(int, int) {
    Table t_receipt = openTable("t_receipt");
    Condition condition = t_receipt.newCondition();
    condition.EQ("debteeName", debtee);
    condition.EQ("debtorName", debtor);
    int count = t_receipt.remove(key, condition);
    emit removeR(count);
    return (count, int(0));
}

function updateReceipt(string debtor, string debtee, int num) public returns(int, int) {
    Table t_receipt = openTable("t_receipt");
    Condition condition = t_receipt.newCondition();
    condition.EQ("debteeName", debtee);
    condition.EQ("debtorName", debtor);

    Entry entry = t_receipt.newEntry();
    entry.set("amount", num);

    int count = t_receipt.update(key, entry, condition);
    emit UpdateR(count);

    return (count, num);
}
```

创建交易，删除交易以及更新交易

```
function selectReceipt(string debtor, string debtee) public returns(string, string, int, ir
    Table t_receipt = openTable("t_receipt");
    Condition cond = t_receipt.newCondition();
    cond.EQ("debteeName", debtee);
    Entries entries = t_receipt.select(key, cond);
    require(entries.size() > 0, "receipt id not exists!");
    Entry entry = entries.get(0);
    int j = -1;
    for (int i = 0; i < entries.size(); i++) {
        entry = entries.get(i);
        if (isStringEQ(debtor, entry.getString("debtorName"))) {
            j = i;
            i = entries.size();
        }
    }
    require(j >= 0, "receipt id not exists!");
    entry = entries.get(j);
    receipt.debteeName = entry.getString("debteeName");
    receipt.debtorName = entry.getString("debtorName");
    receipt.amount = entry.getInt("amount");
    receipt.deadline = entry.getInt("deadline");
    emit findReceipt(debtor, debtee, receipt.amount, receipt.deadline);
    return (debtor, debtee, receipt.amount, receipt.deadline);
}
```

搜索交易

```
function computeAll(string name) public returns (int, int) {
    Table t_receipt = openTable("t_receipt");
    Condition cond = t_receipt.newCondition();
    cond.EQ("debteeName", name);
    Entries entries = t_receipt.select(key, cond);
    Condition cond1 = t_receipt.newCondition();
    cond1.EQ("debtorName", name);
    Entries entries1 = t_receipt.select(key, cond1);
    Entry entry = entries.get(0);
    //selectCompany(name);
    int total = 0;
    for (int i = 0; i < entries.size(); i++) {
        entry = entries.get(i);
        total += entry.getInt("amount");
    }
    entry = entries1.get(0);
    for (i = 0; i < entries1.size(); i++) {
        entry = entries1.get(i);
        total -= entry.getInt("amount");
    }
    return (total + 10000, int(0));
}
```

计算可融资金额

```
function rTransfer(string debtor, string debtee, string toOther, int deadline, int amount1
    Table t_receipt = openTable("t_receipt");
    Condition condition1 = t_receipt.newCondition();
    condition1.EQ("debteeName", debtee);
    condition1.EQ("debtorName", toOther);
    Entries entries1 = t_receipt.select(key, condition1);
    Entry entry = entries1.get(0);
    int amount2 = 0;
    if (entries1.size() == 1) {
        entry = entries1.get(0);
        amount2 = entry.getInt("amount");
        updateReceipt(toOther, debtee, amount1+amount2);
    } else {
        insertReceipt(toOther, debtee, amount1, deadline);
    }
    removeReceipt(debtor, debtee);

    Condition condition2 = t_receipt.newCondition();
    condition2.EQ("debteeName", debtor);
    condition2.EQ("debtorName", toOther);
    Entries entries2 = t_receipt.select(key, condition2);
    if (entries2.size() == 1) {
        entry = entries2.get(0);
        amount2 = entry.getInt("amount");
        updateReceipt(toOther, debtor, amount2-amount1);
    } else {
        insertReceipt(toOther, debtor, amount1, deadline);
    }
}
// (debtor, debtee) --> (toOther, debtee)
function rTransferReceipt(string debtor, string debtee, string toOther, int deadline) priv
    Table t_receipt = openTable("t_receipt");
    Condition condition = t_receipt.newCondition();
    condition.EQ("debteeName", debtee);
    condition.EQ("debtorName", debtor);
    Entries entries = t_receipt.select(key, condition);
    require(entries.size() == 1, "The receipt isnot exist!");
    Entry entry = entries.get(0);
    int amount1 = entry.getInt("amount");

    rTransfer(debtor,debtee,toOther,deadline,amount1);

   //return (int(0), int(0));
}
```

账单转移

```
    function timeGo(string debtor, string debtee, int time) returns(int, int){
        Table t_receipt = openTable("t_receipt");
        Condition condition = t_receipt.newCondition();
        condition.EQ("debteeName", debtee);
        condition.EQ("debtorName", debtor);
        Entries entries = t_receipt.select(key, condition);
        require(entries.size() == 1, "The receipt isnot exist!");
        Entry entry = entries.get(0);
        Entry entry1 = t_receipt.newEntry();
        entry1.set("deadline", entry.getInt("deadline") - time);
        if (entry.getInt("deadline") - time <= 0) {
            entry1.set("amount", int(0));
        }
        int count = t_receipt.update(key, entry1, condition);
        emit IsDeadline(debtor, debtee, entry.getInt("amount"));
        return (entry.getInt("deadline") - time, entry.getInt("amount"));
    }
```

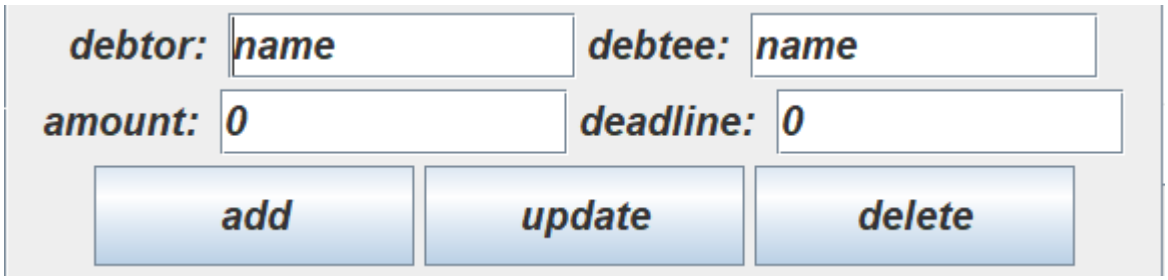
时间是否到期

界面展示

账单表格：

debtor	debtee	amount	deadline
nameA1	nameB1	0	0

创建账单，更新账单，以及删除账单

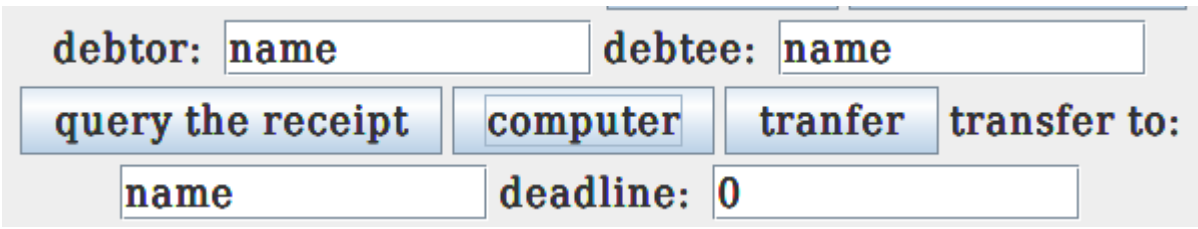


公司注册以及搜索：

公司表格：

company name	money	companyType
Bank		0
company1		1
company2		1
company3		1

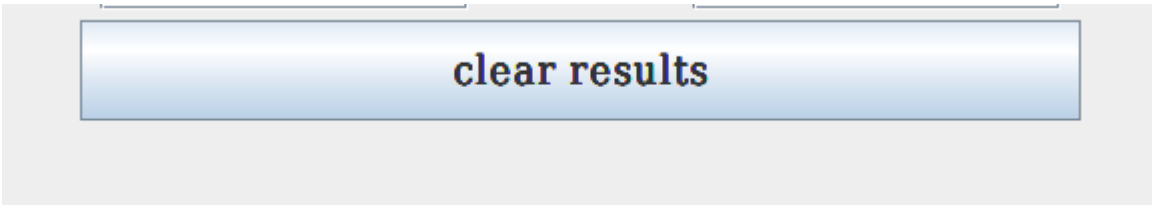
账单搜索：转账，融资



搜索结果显示：

query result	result	result	result	result
company3				
c1	c2	amount:	deadline:	

清除所有搜索结果：



心得体会

这次实验让我对区块链应用有了一个具体的理解，原来半懂不懂的概念比如公钥私钥、账户、节点在区块链应用中扮演什么样的角色也有了了解。

对区块链中的账本，区块，交易，账户，世界状态，共识机制，节点，共识算法，智能合约有了一定的了解，同时通过此次对联盟链的简单使用，对联盟链也相对了解一些。行业里通常将区块链的类型分为公有链，联盟链，私有链。公有链指所有人都可以随时随地参与甚至是匿名参与的链；私有链指一个主体（如一个机构或一个自然人）所有，私有化的管理和使用的链；联盟链通常是指多个主体达成一定的协议，或建立了一个业务联盟后，多方共同组建的链，加入联盟链的成员需要经过验证，一般是身份可知的。正因为有准入机制，所以联盟链也通常被称为“许可链”。