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	2017	专业（方向）	软件工程
学号	17343078	姓名	刘任浩天
电话	null	Email	365498155@qq.com
开始日期	2019/12/07	完成日期	2019/12/13

一、 项目背景

基于已有的开源区 块 链 系 统 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。

二、 方案设计

前端：负责转账，注册，登陆，查询联盟链上所有的企业，查询企业的相关信任额度，查询所有转账记录的UI交互。

后端：采用`Fisco bcos`的Nodejs SDK，处理前端的请求，调用，部署智能合约的相关函数，将结果返回给前端。

链端：编写相关的智能合约。

后端功能展示：

- Nodejs SDK提供的数据库功能具有完备的CRUD功能，且通过查看源码发现其底层通过智能合约实现，因此，只要部署数据库之后，就可以直接使用其功能来完成用户的登陆注册，转账，查看余额等功能。
    - 数据库的部署实现

```
const webapi = require("./nodejs-sdk/packages/api/web3j").Web3jService;
const Configuration = require("./nodejs-
sdk/packages/api/common/configuration").Configuration;

Configuration.setConfig("./nodejs-sdk/packages/cli/conf/config.json");
const utils = require("./nodejs-
sdk/packages/api/common/web3lib/utils");
const { CRUDService, Table, Condition, Entry } = require('./nodejs-
sdk/packages/api');

//查阅Fisco bcos的开发文档找到部署数据库的api
let table = new Table("t_assets", "account", "money,credit", "")

var crud = new CRUDService()
//创建数据库，数据库的名字是t_assets,主键为account，字段为money余额和信用。
crud.createTable(table).then(status => {
    if (status === 0) {
```

```
        return { status: "success", code: status };
    } else {
        return { status: "fail", code: status };
    }
});
```

- 数据库的CRUD操作，取insert插入功能展示

```
//代码的解释写在注释中
//这是数据库CRUD操作中的插入功能实现
if (url_info.pathname == '/insert' && req.method == "POST"){
      var str = "";//接收数据用用
      req.on('data',function(data){
          str += data;
      });
      req.on('end', () => {
          //查询数据库表
          crudService.desc("t_assets").then(tableInfo => {
              var obj = JSON.parse(str)
              //按照查询的表的数据建表，方便后续issert数据
              let table = new Table(tableInfo.tableName, obj.account,
tableInfo.valueFields, tableInfo.optional);
              //取得前端的post请求数据
              let fieldNames = tableInfo.valueFields.split(',');

              let fieldValues = obj["values"].split(',');

              if (fieldNames.length !== fieldValues.length) {

                  throw new Error(`unmatch number of fields, expected $
{fieldNames.length} but got ${fieldValues.length}`);
                }
              let entry = new Entry();
              for (let index in fieldNames) {
                  entry.put(fieldNames[index], fieldValues[index]);

              }
              //插入数据到t_assets数据库表
              crudService.insert(table, entry).then(value=>{
                  console.log(value) if(value == "1"){
                      console.log("insert successfuuly") } });

          });
      })
  }
```

- 前端post请求格式:

```
//前端post请求体，插入数据account id以及余额，信用额度
var requestData2={
    account : "msb",
    values : "200,100"
};


var http = require('http');
```

```
var qs = require('querystring');

var content = JSON.stringify(requestData2);

var options = {
    hostname: '127.0.0.1',
    port: 3000,
    path: '/select',
    method: 'POST',
    headers: {
        'Content-Type': 'application/x-www-form-urlencoded; charset=UTF-8'
    }
};

var req = http.request(options, function (res) {
    //console.log('STATUS: ' + res.statusCode);
    //console.log('HEADERS: ' + JSON.stringify(res.headers));
    res.setEncoding('utf8');
    res.on('data', function (chunk) {
        //console.log('BODY: ' + chunk);
    });
});

req.on('error', function (e) {
    //console.log('problem with request: ' + e.message);
});

// 将数据写入请求体
req.write(content);

req.end();
```

链端功能展示:

- 数据库智能合约代码

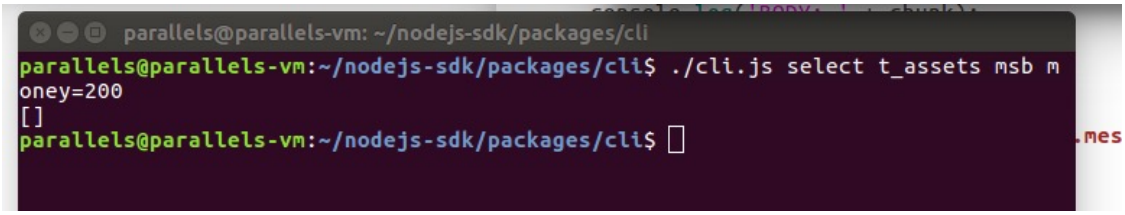
```
pragma solidity ^0.4.24;

import "./Table.sol";

contract Asset {
    event RegisterEvent(int256 ret, string account, uint256 asset_value);
    event TransferEvent(int256 ret, string from_account, string to_account,
uint256 amount);

    constructor() public {
        createTable();
    }

    function createTable() private {
        TableFactory tf = TableFactory(0x1001);
        tf.createTable("t_asset", "account", "asset_value");
    }

    function openTable() private returns(Table) {
        TableFactory tf = TableFactory(0x1001);
```

```
        Table table = tf.openTable("t_asset");
        return table;
    }

    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")));
        }
    }

    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;
    }

    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;
    }
}
```

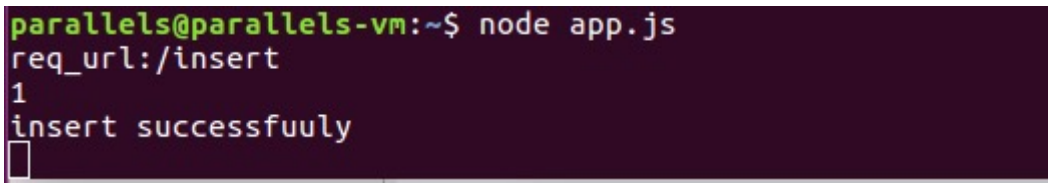
三、功能测试&界面展示

- 部署数据库之后，通过cli查看数据库信息，显示没有属于`msb`用户的信息

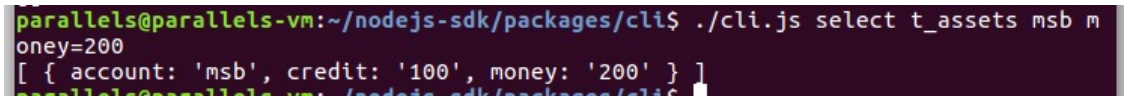


- 前端发出post请求，插入一条`msb`账户的信息，余额是200，信用额度是100

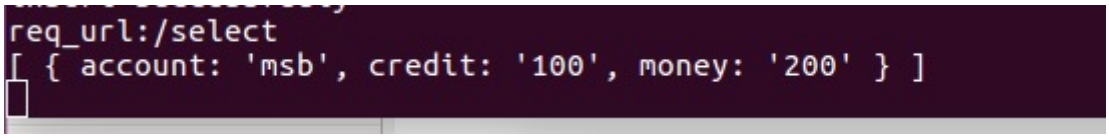
- 服务器查看请求，插入成功



- `CLI`界面确认插入结果，插入成功



- 前端发出查询请求，查询的关键字是account，服务器正常返回结果

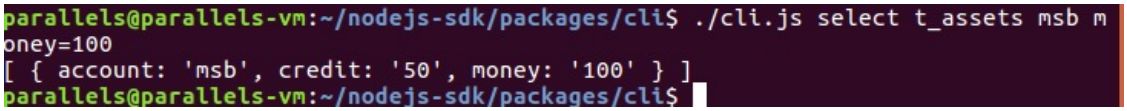


- 前端发出update请求，将`msb`账户余额改为100，信用额度为50

```
var requestData2={
    account : "msb",
    values : "100,50"
};
```

- 服务器正常返回结果，值为1表示update成功

- `CLI`界面查看是否更新成功，确认查询成功



- 由前端根据交易信息，生成CRUD操作。比如转账就是两次update操作更新接受双方的余额，信用额度等。由于Nodejs SDK数据库底层实现就是智能合约，所以交易会自动上链，但是我前端不太会，本次大作业没有前端。

四、实验心得

通过这次实验，我收获了很多区块链实践的知识，这也是这门课这学期最大的收获了。去开连任何节点都可以创建交易，在经过一段时间的确认之后，就可以合理地确认该交易是否为有效，区块链可有效地防止双花问题的发生。总之，区块链这个新兴技术一定会有很大的发展。