**《数据科学案例综合实训》实验报告**

实验序号： 实验项目名称：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 学　　号 | 2010716318 | 姓　　名 | 黄飞阳 | 专业、班 | 20大数据1班 | |
| 实验地点 | 精工1-214 | 指导教师 | 王琰 | 实验时间 | 2022/7/3 | |
| 1. 实验目的及要求   1、理解和掌握数据库中角色授权的实现方法  2、Linux下，用C语言实现角色授权的过程。  二、实验设备（环境）及要求  1、操作系统: Linux  2、参考的开源数据库管理系统：PostgreSQL数据库   1. 实验内容与步骤：   题目链接：<http://118.190.20.162/view.page?gpid=T146>      四、分析与讨论  当我拷贝传参的时候40分，当我用哈希表的时候70分，当我引用哈希传参的时候，100分，然而我的算法根本没变。 | | | | | | |
| 五、教师评语  签名：  日期： | | | | | | 成绩 |

附源程序清单：

#include <iostream>

#include <string>

#include <unordered\_map>

#include <algorithm>

#include <unordered\_set>

using *namespace* std;

*// 角色*

*struct* Role {

*// 角色名*

    string roleName;

*// 操作是否有 \**

*bool* optionHasStar;

*// 种类是否有 \**

*bool* categoryHasStar;

    Role() {

        optionHasStar = false;

        categoryHasStar = false;

    }

*// 操作清单*

    unordered\_set<string> operations;

*// 资源种类清单*

    unordered\_set<string> categories;

*// 资源名清单*

    unordered\_set<string> resNames;

*bool* is\_option\_on\_resource\_Valid(*const* string *&operation*, *const* string *&category*, *const* string *&resName*) *const* {

*// 会快吧,1ms*

        if (!*this*->optionHasStar && !operations.count(operation)) {

            return false;

        }

        if (!*this*->categoryHasStar && !categories.count(category)) {

            return false;

        }

        if (!*this*->resNames.empty() && !resNames.count(resName)) {

            return false;

        }

        return true;

    }

};

*// 根据用户名获取关联的用户*

unordered\_map<string, unordered\_set<string>> roleMap;

*// 根据组名获取组内用户*

unordered\_map<string, unordered\_set<string>> groupMap;

*bool* check\_operation(*const* string *&userName*,

                     unordered\_map<string, Role> *&userInfo*,

*const* vector<string> *&groupNames*,

*const* string *&operation*,

*const* string *&category*,

*const* string *&resName*

) {

    Role &needCheck = userInfo[userName];

*// 自己能不能执行*

    if (needCheck.is\_option\_on\_resource\_Valid(operation, category, resName)) {

        return true;

    }

*// 有没有直接跟该用户关联的，而且可以执行该操作的*

*// roleMap[userName] : 关联 userName 的用户*

    for (*const* *auto* &roleName:roleMap[userName]) {

        if (userInfo[roleName].is\_option\_on\_resource\_Valid(operation, category, resName)) {

            return true;

        }

    }

*// 关联的用户组里面，有没有谁是可以执行该操作的*

*// groupMap[groupName] : 在 groupName 中的用户*

    for (*const* string &groupName:groupNames) {

        for (*const* *auto* &role:groupMap[groupName]) {

            if (userInfo[role].is\_option\_on\_resource\_Valid(operation, category, resName)) {

                return true;

            }

        }

    }

    return false;

}

*int* main() {

*int* n, m, q;

    cin >> n >> m >> q;

    unordered\_map<string, Role> userInfo;

*// 角色*

    for (*int* i = 0; i < n; i++) {

        string userName;

        cin >> userName;

        userInfo[userName] = Role();

        Role &role = userInfo[userName];

        role.roleName = userName;

*int* n\_option;

        cin >> n\_option;

        for (*int* j = 0; j < n\_option; j++) {

            string option;

            cin >> option;

            if (option == "\*") {

                role.optionHasStar = true;

                continue;

            }

            role.operations.insert(option);

        }

*int* n\_category;

        cin >> n\_category;

        for (*int* j = 0; j < n\_category; j++) {

            string category;

            cin >> category;

            if (category == "\*") {

                role.categoryHasStar = true;

                continue;

            }

            role.categories.insert(category);

        }

*int* n\_resource\_name;

        cin >> n\_resource\_name;

        for (*int* j = 0; j < n\_resource\_name; j++) {

            string resName;

            cin >> resName;

            role.resNames.insert(resName);

        }

    }

*// 关联*

    for (*int* i = 0; i < m; ++i) {

        string userName;

        cin >> userName;

*int* n\_union;

        cin >> n\_union;

        for (*int* j = 0; j < n\_union; j++) {

            string type;

            string userOrGroup;

            cin >> type >> userOrGroup;

            if (type == "u") {

                roleMap[userOrGroup].insert(userName);

            } else {

                groupMap[userOrGroup].insert(userName);

            }

        }

    }

*// 检查*

    for (*int* i = 0; i < q; ++i) {

        string userName;

        cin >> userName;

        userInfo[userName] = Role();

        Role &role = userInfo[userName];

        role.roleName = userName;

*int* n\_group;

        cin >> n\_group;

        vector<string> groupNames;

        for (*int* j = 0; j < n\_group; j++) {

            string groupName;

            cin >> groupName;

            groupNames.emplace\_back(groupName);

        }

        string option, category, resName;

        cin >> option >> category >> resName;

        if (check\_operation(userName, userInfo, groupNames, option, category, resName)) {

            cout << 1 << "\n";

        } else {

            cout << 0 << "\n";

        }

        userInfo.erase(userName);

    }

    return 0;

}