西安交通大學

XI'AN JIAOTONG UNIVERSITY

本科生课程报告

基于 Java 事件处理机制的 RBAC0 权限模型实现

课程： 软件系统设计与分析

姓名： 杨豪

班级： 软件 2101

时间： 2022 年 10 月

摘要

在 Java 语言中，当用户与 GUI 组件交互时，GUI 组件能够激发一个相应事件。例如，用户按动按钮、滚动文本、移动鼠标或按下按键等，都将产生一个相应的事件。Java 提供完善的事件处理机制，能够监听事件，识别事件源，并完成事件处理。RBAC 模型即基于角色的访问控制，是一种很有效的权限分配模型。本文从 RBAC 模型簇中最基本的模型 RBAC0 的概念开始，介绍了用 PowerDesigner 设计 RBAC0 并将其和 Java 的事件处理机制联系的实现方法。最后比较并分析了 RBAC 模型簇内常见模型的特点。

关键词：事件处理机制; RBAC.

目录

1 RBAC0 模型概述

1.1 RBAC 模型思想

RBAC（Role-Based Access Control）是基于角色的访问控制。在 RBAC 中，权限与角色相关联，而用户通过成为适当角色的成员而得到这些角色的权限，这就极大地简化了权限的管理。这样管理都是层级相互依赖的，权限赋予给角色，而把角色又赋予用户，这样的权限设计很清楚，管理起来很方便。

虽然可以直接给用户分配权限，只是直接给用户分配权限，少了一层关系，扩展性弱了许多，适合那些用户数量、角色类型少的平台。对于通常的系统，比如：存在多个用户拥有相同的权限，在分配的时候就要分别为这几个用户指定相同的权限，修改时也要为这几个用户的权限进行一一修改。有了角色后，我们只需要为该角色制定好权限后，将相同权限的用户都指定为同一个角色即可，便于权限管理。对于批量的用户权限调整，只需调整用户关联的角色权限，无需对每一个用户都进行权限调整，既大幅提升权限调整的效率，又降低了漏调权限的概率。

1.2 RBAC0 模型

RBAC 模型可以分为：RBAC0、RBAC1、RBAC2、RBAC3 四种。其中 RBAC0 是基础，也是核心的底层逻辑，RBAC1、RBAC2、RBAC3 都是以 RBAC0 为基础的升级。

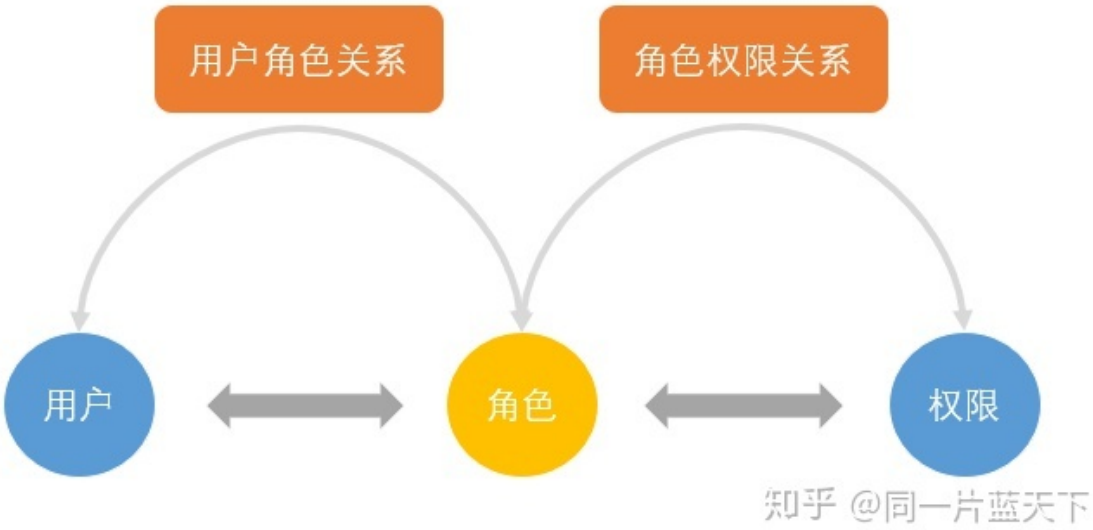

图 1: RBAC0 模型思路

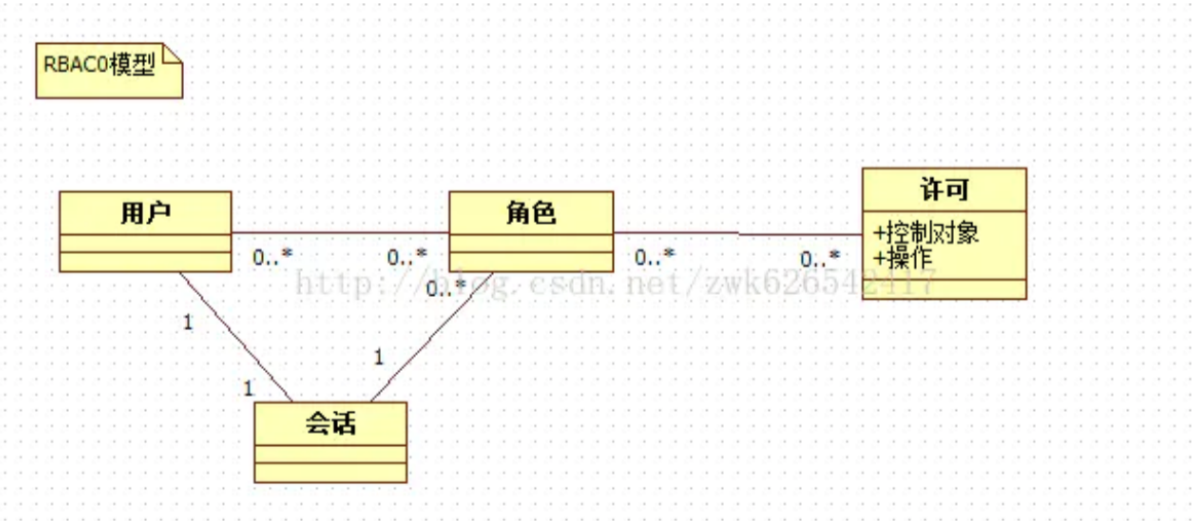

图 2: RBAC0 数据库表图

2 RBAC0 模型的实现

下面具体讲述如何将 Java 中的 Action 事件处理机制和 RBAC0 模型相结合并实现，此处参考西安交通大学教务平台的成绩公布系统设计了一个大学生成绩公布系统，并实现了 RBAC0 模型的权限控制方法。

2.1 数据库的建立

参照 RBAC0 模型的概念，利用 PowerDesigner 设计得到概念模型 UML 图如下

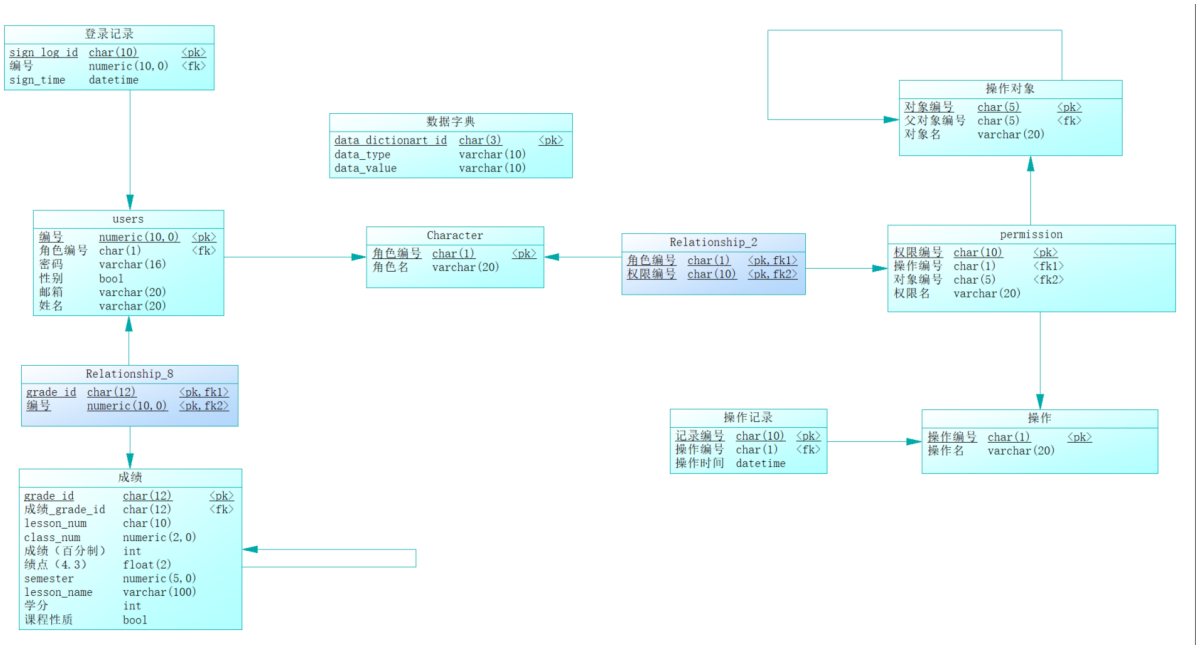

图 3: 基于 RBAC0 的学生成绩管理系统的概念模型

通过 PowerDesigner 内置的工具转换物理模型如下

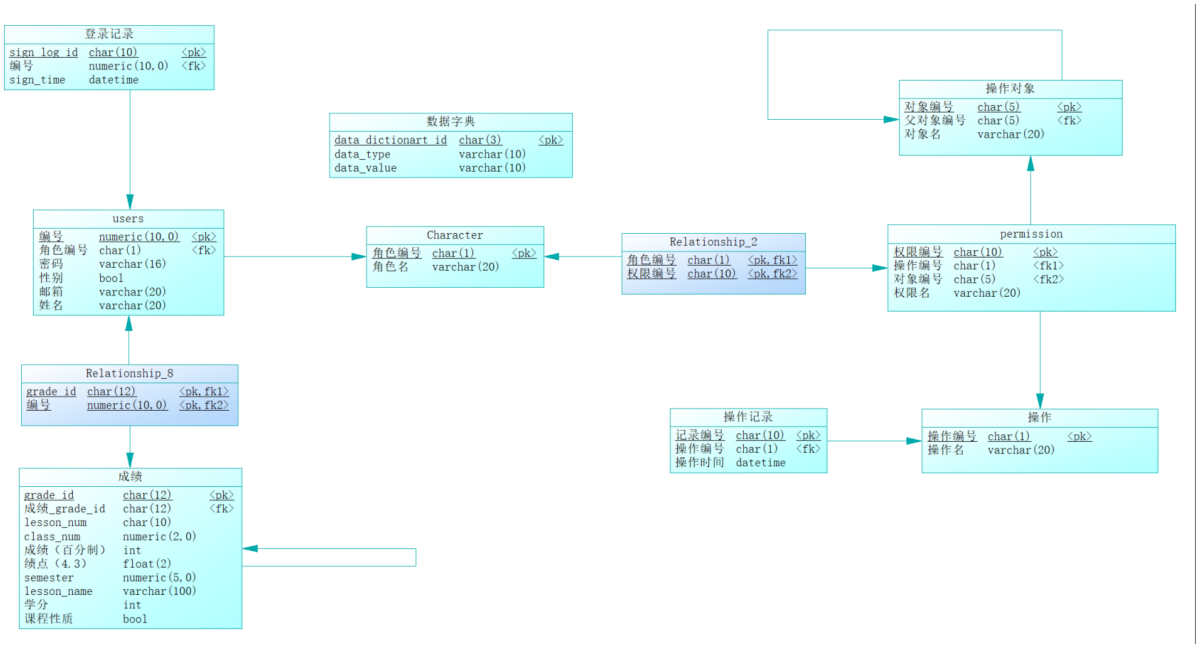

图 4: 基于 RBAC0 的学生成绩管理系统的物理模型

并通过 PowerDesigner 内置的工具转换得到 sql 代码。

2.2 GUI 的搭建

因为任务要求使用 Action 事件处理机制，所以首先需要用 JavaSwing 搭建一个 GUI。

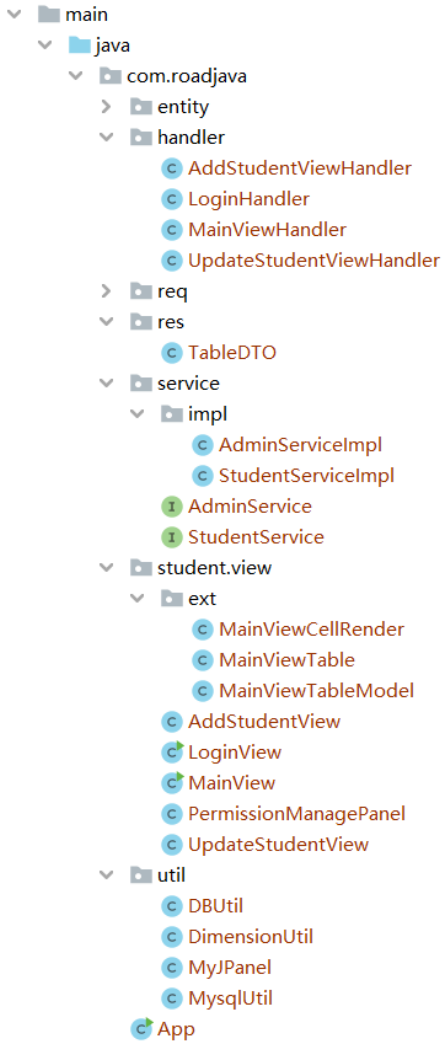

图 5: 初始登陆界面

图 6: 初始登陆界面

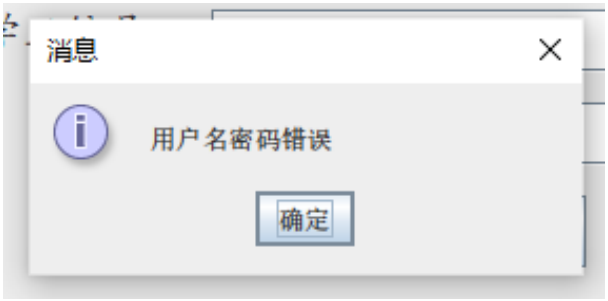

图 7: 输入空

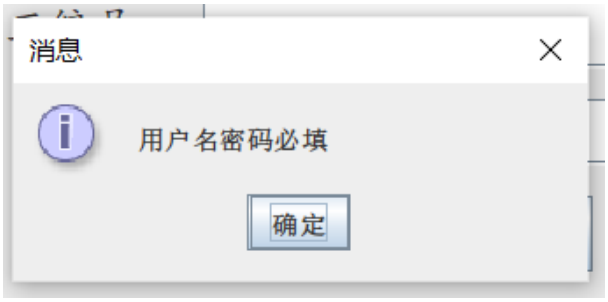

图 8: 密码错误

图 9: 主界面

图 10: 添加界面

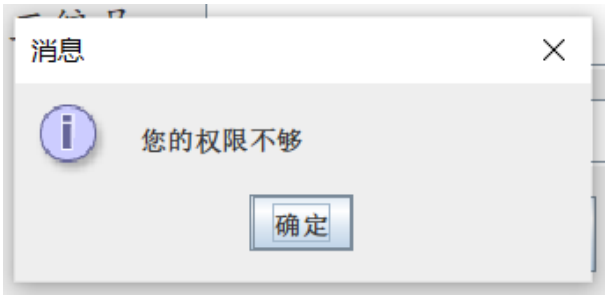

图 11: 违权界面

2.3 Action 事件和 RBAC 权限管理相联系

下面要做的就是将主界面的这些 button 添加 rbac 模型权限的 ActionListener

Listing 1: AddPermissionPanel.java L50-69

```
1
2  // 为按钮设置监听器
3  actionButton.addActionListener(new AbstractAction() {
4      @Override
5      public void actionPerformed(ActionEvent e) {
6          try {
7              Statement statement = connection.createStatement();
8              Map<String, String> map = new HashMap<>();
9              map.put("action_id", MysqlUtil.getNextId("t_action", "action_id
                   "));
10             map.put("action_name", actionText.getText());
11             // 将该操作添加到数据库中
12             String insertObj = MysqlUtil.Insert("t_action", map);
13             statement.executeUpdate(insertObj);
14
15             //返回结果
16             new ResultDialog("Success!");
17
18         } catch (SQLException ex) {
```

```
19              ex.printStackTrace();
20          }
21      }
22  });
```

Listing 2: MainPanel.java L132-150

```
1
2   // 获取当前用户的 权限并disable菜单栏的对应按钮
3   try {
4       Statement statement = connection.createStatement();
5       String getPermissionSql = "select permission_name from t_user_info,
            t_permission,t_role_permission where info_id=" + info_id + " and
            t_user_info.role_id=t_role_permission.role_id and t_role_permission
            .permission_id=t_permission.permission_id";
6       ResultSet rs = statement.executeQuery(getPermissionSql);
7       Set<String> permissionSet = new HashSet<>();
8       while (rs.next()) {
9           permissionSet.add(rs.getString(1));
10      }
11      System.out.println(permissionSet.toString());
12      // 禁用该用户没有的权限
13      for (MyMenuItem menuItem : itemList) {
14          if (!permissionSet.contains(menuItem.getPermission())) {
15              menuItem.setEnabled(false);
16          }
17      }
18  } catch (SQLException e) {
19      e.printStackTrace();
20  }
```

Listing 3: PermissionManagePanel.java L196-270

```
1
2  public void setMakePermissionTable() {
3
4      makePermissionTable.removeAll();
5
6      // 获取所有操作对象列表
7      Obj[] objList = getObj();
8
9      // 获取所有操作动作列表
10     Action[] actionList = getAction();
11
12     // 获取已经分配好的所有权限
13     Permission[] permissionList = getPermission();
14
15     // 为设置表格设置样式
16     makePermissionTable.setLayout(new GridLayout(0, objList.length + 1, 10,
           3));
17     makePermissionTable.setBounds(20, 110, 700, (actionList.length + 1) *
           30);
18
19     // 第一行为所有的操作对象
20     makePermissionTable.add(new JLabel()); //占位
21     for (Obj obj : objList) {
22         makePermissionTable.add(new JLabel(obj.objName));
23     }
24
25     // 初始化选择框并添加监听对象,在监听到勾选状态发生改变时上传数据到数据
           库
26     checkBoxes = new JCheckBox[actionList.length][objList.length];
27     for (int i = 0; i < actionList.length; i++) {
```

```
28          makePermissionTable.add(new JLabel(actionList[i].actionName));
29          for (int j = 0; j < checkBoxes[i].length; j++) {
30              checkBoxes[i][j] = new JCheckBox();
31              makePermissionTable.add(checkBoxes[i][j]);
32              Obj obj = objList[j];
33              Action action = actionList[i];
34              JCheckBox checkBox = checkBoxes[i][j];
35              checkBox.addActionListener(new AbstractAction() {
36                  @Override
37                  public void actionPerformed(ActionEvent e) {
38                      String objId = obj.objId;
39                      String objName = obj.objName;
40                      String actionId = action.actionId;
41                      String actionName = action.actionName;
42                      try {
43                          Statement statement = connection.createStatement();
44                          String changePermissionSql;
45                          Map<String, String> map = new HashMap<>();
46                          map.clear();
47                          map.put("obj_id", objId);
48                          map.put("action_id", actionId);
49
50                          if (checkBox.isSelected()) {
51                              // 新增对应的权限
52                              String permission_id = MysqlUtil.getNextId("
                                    t_permission", "permission_id");
53                              map.put("permission_id", permission_id);
54                              map.put("permission_name", actionName + objName
                                    );
55                              changePermissionSql = MysqlUtil.Insert("
```

```
                                t_permission", map);
56                          } else {
57                              // 删除对应的权限
58                              changePermissionSql = MysqlUtil.Delete("
                                    t_permission", map);
59                          }
60                          statement.executeUpdate(changePermissionSql);
61
62                      } catch (SQLException ex) {
63                          ex.printStackTrace();
64                      }
65
66                      System.out.println(actionName + objName);
67                  }
68              });
69          }
70      }
71
72      // 设置默认选中的权限组合
73      for (Permission permission : permissionList) {
74          checkBoxes[Integer.parseInt(permission.actionId) - 1][Integer.
                parseInt(permission.objId) - 1].setSelected(true);
75      }
76  }
```

3 RBAC 模型簇的分析与比较

- RBAC0 模型：最基础最简单的 RBAC 思想
- RBAC1 模型：基于 RBAC0 模型，增加了子角色，引入了继承概念，即子角色可以继承父角色的所有权限

- RBAC2 模型：基于 RBAC0 模型，增加了对角色的一些限制：
  - 角色互斥：同一用户不能分配到一组互斥角色集合中的多个角色，互斥角色是指权限互相制约的两个角色。
  - 基数约束：一个角色被分配的用户数量受限。
  - 先决条件角色：指要想获得较高的权限，要首先拥有低一级的权限。
  - 运行时互斥：例如，允许一个用户具有两个角色的成员资格，但在运行中不可同时激活这两个角色。
- RBAC3 模型：综合了 RBAC0、RBAC1 和 RBAC2 的所有特点。

	特点	适用场景
RBAC0	最基础	普通无限制场景
RBAC1	RBAC0 + 继承（等级）	层级分明的管理人员权限分配
RBAC2	RBAC0 + 约束（职责分离）	存在互斥的职位
RBAC3	RBAC1 + RBAC2	同时符合 RBAC1 和 RBAC2 的场景

4 总结与回顾

本题目可以说是本学期开学以来所有科目中对我挑战性最大的一个题目，因为我并没有 java GUI、DBAC、mysql 和 PowerDesigner 的操作基础，可以说除了 RBAC 的理论知识完全零基础。

在我写代码的过程中 PowerDesigner 的工具套件对我帮助最大，概念模型转物理模型再转 sql 源码的功能大大加快了我的进度。