基于角色的权限管理可重用解决方案①

Reusable Total Solution of Role – Based Privilege Management

刘鹏远 （湖北经济学院 计算机学院 湖北武汉 430205）

摘要：本文基于作者曾开发的一个项目，将基于角色的权限管理部分抽取出来划分为数据库分析设计、权限菜单处理、角色修改与查看、用户角色识别以及功能点按钮处理五大部分进行阐述，形成了一个总体解决方案。该解决方案与具体语言和应用系统框架无关，具备良好的通用性，列出的一些核心算法在系统提交后能满足要求。

关键词：角色权限管理 信息管理系统

本文借鉴 struts menu 的思想，将权限做成菜单（菜单的动作就是出来的页面，功能点则对应该页面上的按钮动作），这样在系统分析阶段由分析员给出静态页面原型后就自然完成了权限的划分，接着进行角色定义过渡非常自然。该解决方案包括业务逻辑分析设计数据库、定义菜单和设计权限和角色、设计访问数据库 DAO、新增/修改/查看角色时的展现、身份识别登录、功能点检查等一系列过程，不依赖任何开发包和应用框架，以下将对其中几个关键的技术难点进行展开。

1 数据库分析设计

分析员了解需求后定义系统原型（包括菜单、按钮和各种业务数据的静态页面示例）提交用户评审，用户评审通过后，将在原型基础上进行后继的数据库分析设计[3]。使用存储在数据库中的表记录来展现需要动态变化的内容（菜单，按钮等）[4]，该解决方案非常重要一步是如何设计数据库表来表达用户、角色、权限以及功能点的关系。

1.1 权限和功能点静态表

如果将菜单项设计为权限，意味着若用户具有某个权限登录后就能点击该菜单项。菜单的层次性决定了权限表是具有递归层次的树状结构，即除了根节点，表中其它节点都有 ParentNo 字段指向父节点 MenuNo。更进一步，如果需要精细控制页面上按钮，比如对页面某个按钮系统管理员允许点击保存而班主任点击保存却提示没有权限。按钮也需要动态控制，使用功能点表来保存。两个静态表定义了系统中需要控制的所有权限和功能点，该项任务安排在系统原型确定后立即进行。

1.2 角色分析

对系统用户建立用户表，对用户类型做角色分析使用户和权限不直接耦合（这样能动态调整用户访问权限和功能点控制权限，使系统更有灵活性）。一个用户可有多个角色（比如张三是系统管理员又是班主任，这两个角色有不同的权限集合），而一个角色也可被分配给多个用户，用户与角色的多对多关系使用用户角色关联转化为两个多对一关系。一个角色，对应着多项权限，而一个权限，可被分配给多个角色，角色与权限的多对多关系使用角色权限关联转化为两个多对一关系。

上面几个表的设计，能够完成用户查找角色，角色查找权限获得自己授权的操作（菜单）。如果需要进一步细粒度控制页面上的按钮（即功能点控制），还需要定义角色权限功能点表它定义每个角色具有的权限和功能点（两个角色，有可能有着相同权限但不同的功能点）。

2 权限菜单处理

用户登录时查用户表，验证通过后查用户角色表得角色编号集合，由角色编号集合查角色权限表得权限编号集合（这就是该用户具有的权限编号），再继续关联查权限表得该用户具有的权限集合。考虑到一用

① 基金项目：湖北省教育厅重点项目（B200619001）

户具有多个角色，那么其权限应该是这些不同角色查找到的权限集合的并集（比如张三有咨询员和班主任两个角色，张三登录系统后就具有咨询员和班主任这两个角色都有的权限）。

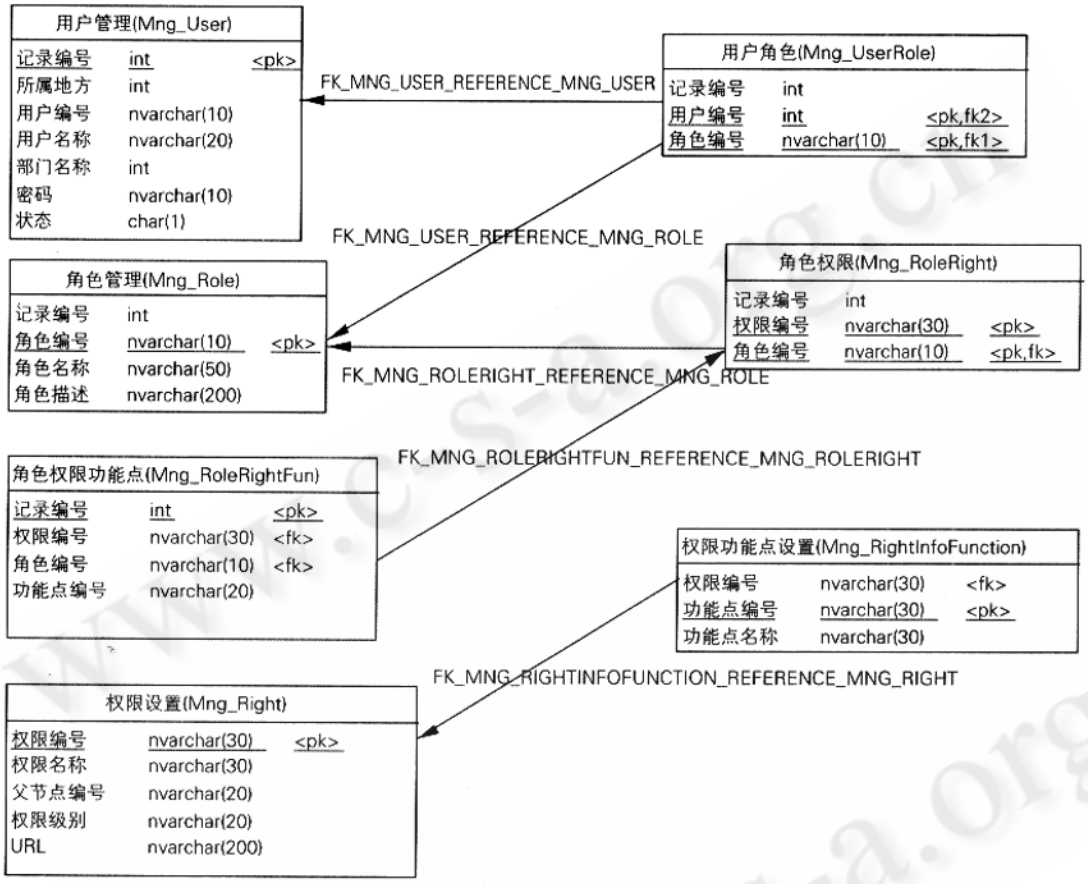

图 1　角色权限管理系统七张表的设计

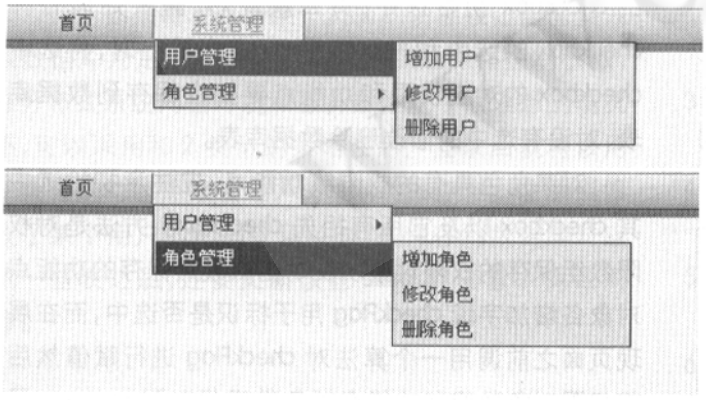

图 2　菜单串生成的菜单示例

2.1 菜单生成

使用 javascript 技术，对 menu 字符串传入一 js 函数 showmenu(menu)[5] 就生成如图 2 所示的菜单：

menu ="" +"首页;;系统管理;;用户管理;;增加用户;;修改用户;;删除用户;;角色管理;;增加角色;;修改角色;;删除角色::1-000;;1-001;;1-001-002;;1-001-002-003;;1-001-002-004;;1-001-002-005;;1-001-006;;1-001-006-007;;1-001-006-008;;1-001-006-009::1;;1;;2;;3;;3;;3;;2;;3;;3;;3::/1.jsp;;;;;;/2.jsp;;/3.jsp;;/4.jsp;;;;/5.jsp;;/6.jsp;;/7.jsp "+"";

2.2 递归调整读出的具有的权限集合

权限表和功能点表的内容随着开发推进可能需要修改，而权限表表结构带有明显的层次性，对它的修改将需要全部删除记录后再按照深度顺序重新录入，将异常繁琐。菜单的层次性要求按照从上到下从左到右深度优先[6-7]的顺序拼接权限表记录，然后将串传入 showmenu 形成菜单，如 2.1 节图 2 所示。展现一个角色具有的权限和功能点同样要求保持层次性（父级权限应该在子级权限或功能点的上一行，子级权限或功能点应该比上一层更缩进）。如果系统定义了动态调整权限和功能点的菜单接口，这也需要按深度优先顺序调整权限表记录顺序。

总之，若对权限表记录不做深度优先序排列，在生成菜单和增删改角色的页面展现时会异常困难，下面将给出调用 getDFSMenuNo 算法调整权限数

组顺序的流程。注意调用前先虚构一根权限做树的根，传入一 ArrayList vRight 代表该用户的权限数组，并保存一 ArrayList vOutRight 返回调整为深度优先序的权限数组。步骤如下：

(1) 首先将虚构的根权限加入到权限数组 vRight；

MngRight mngRightRoot = new MngRight ("000" , "root" , "root" , "" , "0") ;

ArrayList vRight = new ArrayList ( ) ; vRight. add (mngRightRoot) ;

(2) 将访问数据库得到的权限结果集加入到 vRight；

…vRight. add(…) ;…

(3) 调用 getDFSMenuNo 调整传入的 vRight 权限数组为深度优先序保存至 vOutRight：

ArrayList vOutRight = new ArrayList ( ) ; getDFS-MenuNo(0,vRight,vOutRight) ;

当最外层递归遍历处理完 vRight 权限数组时，也就是处理完虚构的根权限的所有儿子时(顶级权限)时，主函数的调用 getDFSMenuNo 结束。当权限表记录规模为 N 时，每个节点都需要遍历数组找到儿子，向上递归获得祖先前缀，然后再向下深度递归处理子孙。因此最坏情况下算法复杂度为 $O(N^3)$。当部署环境的服务器设备性能或分布式网络环境不尽理想时，考虑到最终使用的都是带前缀的权限记录，可在输入静态表的权限记录时就带上其祖先前缀——这样省去了向上递归获得祖先前缀的过程算法复杂度改善为 $O(N^2)$。

3 角色修改与查看

调整权限数组为深度优先序后，vOutRight 就是展现前的数据准备，但在算法上的准备上还很不够，多层次 checkbox 下各种选中与去除选中的级联控制控制逻辑相当复杂。见图 3 -7。

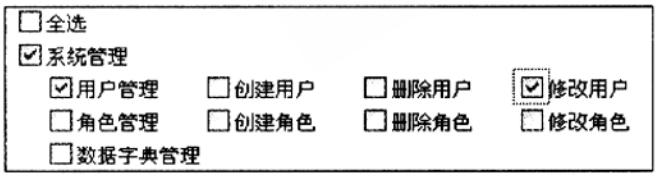

图 3　选中一功能点其所有父亲和祖先权限被选中

在新建/修改/查看角色时需要按深度优先序展现出系统定义的所有权限和功能点(权限表和功能点表的所有记录)，并将该角色具有的权限和功能点对应的 checkbox 初始选中；页面展现后，需要处理各种 checkbox 的选中和去除选中；当点击确定时，将选中 checkbox 的对应权限和功能点要相应保存到数据库表，对没有选中的修改删除数据库表。

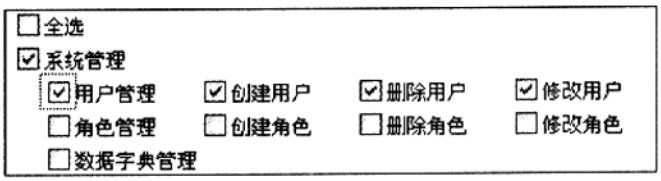

图 4　选中一权限其所有子孙权限和功能点被选中

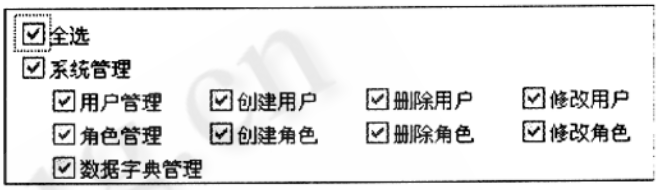

图 5　全选控制

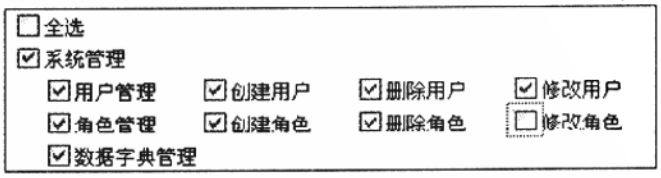

图 6　一功能点去除选中全选框被去除选中

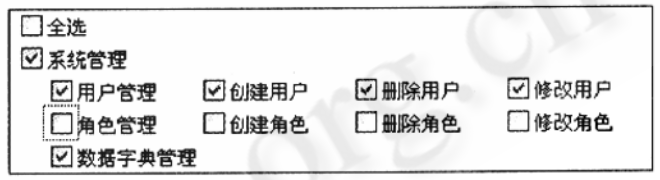

图 7　一权限去除选中其子孙权限和功能点被去除选中

对某角色具有的权限或功能点，应注意初始选中其 checkbox 以及它所有祖先 checkbox。方法是对权限数组保存的权限对象以及功能点数组保存的功能点对象各增加字段 checkFlag 用于标识是否选中，而在展现页面之前调用一个算法对 checkFlag 进行赋值然后将权限对象数组和功能点对象数组保存到 request，展现的页面取 request，迭代取两个数组的元素并置 checkbox 初始状态。

对选中和去除选中的控制需编写脚本语言将递归

与遍历结合,这样要对 checkbox 的 name 做层次命名安排。可以将 name 定义为带上祖先权限编号前缀(多级祖先前缀以"-"间隔)的对应权限编号或功能点编号。角色权限功能点表只保存选中的叶子级最后一级权限和对应具有的功能点,而角色权限表要保存选中的权限以及它的所有祖先权限。因此对点击确定后的处理,需要区分权限和功能点,为此对 checkbox 的 name 定义加上一条约定——功能点编号 checkbox 的 name 以."/"结尾表示是功能点。对于如何识别选中的 checkbox 这里巧妙地使用了许多同 name 的 hidden 字段来跟踪对应的 checkbox 的选中状态,以方便后继的保存和修改处理。

4 用户角色识别

登录系统时,要验证三件事情:

(1) 判定是否是合法用户。

读用户表,用户名/密码匹配。

(2) 查找该用户具有的权限集合(菜单),并调整为深度优先顺序,为菜单做准备。

合法用户登录以后就会出现他具有权限的对应菜单。菜单是每次进入新页面都要重新调用 js 中的 showmenu 函数生成菜单的,如果每次进入新页面都重新读数据库多张表得到权限集合以及做调整顺序,显然耗费太高。使用这样的策略:在登录后第一次得到该用户的权限集合时将该 menu 串写入 session。以后进入新页面都 include 包含一个调用 showmenu(menu)的页面。

(3) 如果每次点击某个页面的功能按钮都需要去读数据库判定他是否具有该功能点的操作控制权限,在分布式系统网络环境和服务器性能不很理想的情况下,可以采用和 2 类似的策略在用户登录后就将该用户具有的功能点集合写入 session 保存[5]。注意当修改了某个用户的密码或角色时,在保存到数据库对应的一些表之后,还要更新该些 session 变量以保证同步。

5 功能点按钮处理

功能点的控制权限即对应着按钮的点击处理,只有最后一级的叶子权限才有对应的功能点。由于权限直接对应菜单,用户具有该权限才能看到该菜单项,才有点击该菜单项的可能。点击以后就跳转到出来的目标页面,而该页面上的按钮则对应着功能点。为此定义一个功能函数 checkFunNo 放在基类中,各个点击按钮引发动作的类都应继承该类,嵌入先调用该判断函数,如果具有该权限,就继续,否则跳转到一个错误页面。该函数要求传入一个 request,以及一个静态字符串,实际上后者就是该用户动作的按钮的"名称",对应着功能点静态表中记录的功能点编号。客户端页面点击按钮的动作应该告诉 checkFunNo 按下的是哪个按钮,然后 checkFunNo 进行判断是否具有该功能点的控制权限。

6 小结

本文将角色权限管理技术划分为五个方面的技术难点,分而治之给出详尽的解决方法,形成了具备相当可重用性且不依赖任何应用系统框架的角色权限管理系统总体解决方案。本解决方案已在作者主持开发的省教育厅重点项目中得到检验,成熟稳定。由于该方案涉及算法较多,算法异常复杂,限于篇幅本文只能在宏观角度上给出梗概方案框架,继续的细节,有待后继发表,有兴趣读者可与本文作者联系。

参考文献

1 Struts - menu 源码分析[EB/OL]. http://www.raibledesigns.com/struts-menu, 2006-9-10.

2 Acegi + Hibernate 动态实现基于角色的权限管理[EB/OL]. http://www.blogjava.net/limq/archive/2006/02/16/28585.html, 2006-02-16.

3 罗晓沛, 侯炳辉, 系统分析员教程[M], 北京: 清华大学出版社, 2004.

4 萨师煊, 王珊, 数据库系统概论(第三版)[M], 北京: 高等教育出版社, 2002.

5 飞思科技产品研发中心, JSP 应用开发详解(第二版)[M], 北京: 电子工业出版社, 2006.

6 王晓东, 算法设计与分析(第一版)[M], 北京: 清华大学出版社, 2003.

7 唐策善, 李龙澎, 黄刘生, 数据结构——用 C 语言描述[M], 北京: 高等教育出版社, 1995.