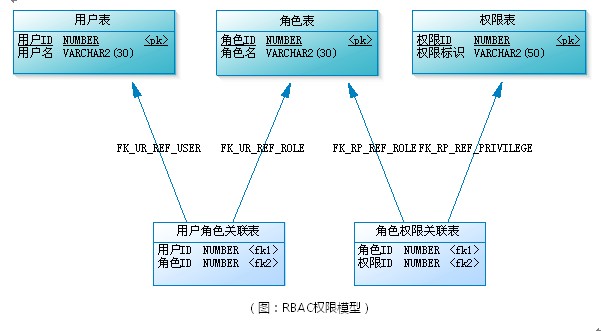
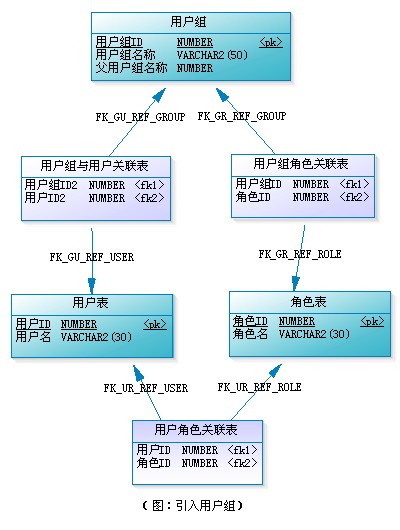
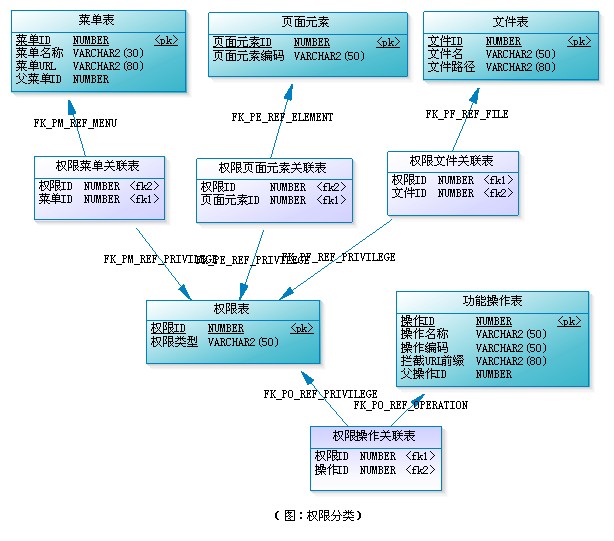
RBAC权限管理

RBAC（Role-Based Access Control，基于角色的访问控制），就是用户通过角色与权限进行关联。简单地说，一个用户拥有若干角色，每一个角色拥有若干权限。这样，就构造成“用户-角色-权限”的授权模型。在这种模型中，用户与角色之间，角色与权限之间，一般者是多对多的关系。（如下图）

  
  
角色是什么？可以理解为一定数量的权限的集合，权限的载体。例如：一个论坛系统，“超级管理员”、“版主”都是角色。版主可管理版内的帖子、可管理版内的用户等，这些是权限。要给某个用户授予这些权限，不需要直接将权限授予用户，可将“版主”这个角色赋予该用户。

当用户的数量非常大时，要给系统每个用户逐一授权（授角色），是件非常烦琐的事情。这时，就需要给用户分组，每个用户组内有多个用户。除了可给用户授权外，还可以给用户组授权。这样一来，用户拥有的所有权限，就是用户个人拥有的权限与该用户所在用户组拥有的权限之和。（下图为用户组、用户与角色三者的关联关系）

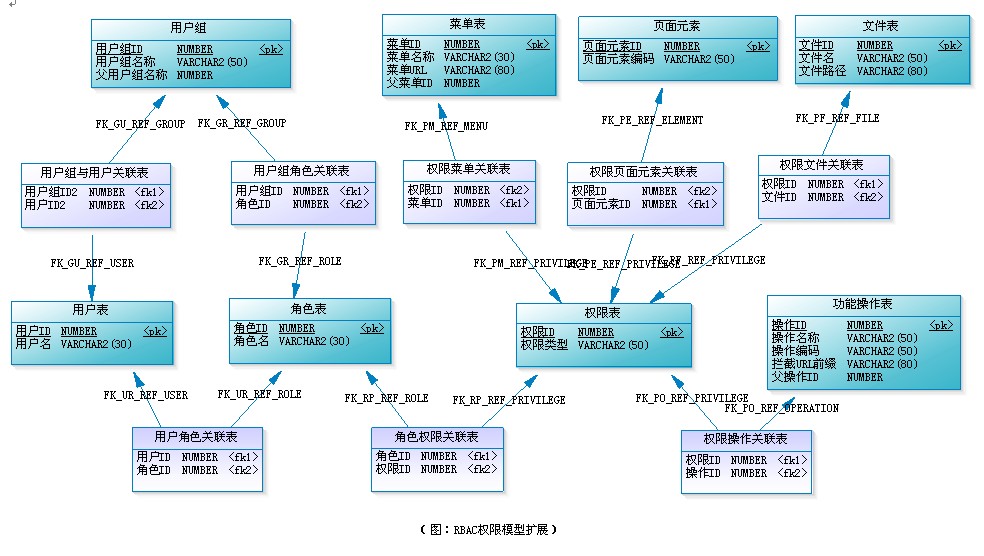


在应用系统中，权限表现成什么？对功能模块的操作，对上传文件的删改，菜单的访问，甚至页面上某个按钮、某个图片的可见性控制，都可属于权限的范畴。有些权限设计，会把功能操作作为一类，而把文件、菜单、页面元素等作为另一类，这样构成“用户-角色-权限-资源”的授权模型。而在做数据表建模时，可把功能操作和资源统一管理，也就是都直接与权限表进行关联，这样可能更具便捷性和易扩展性。（见下图）  
  
请留意权限表中有一列“权限类型”，我们根据它的取值来区分是哪一类权限，如“MENU”表示菜单的访问权限、“OPERATION”表示功能模块的操作权限、“FILE”表示文件的修改权限、“ELEMENT”表示页面元素的可见性控制等。

这样设计的好处有二。其一，不需要区分哪些是权限操作，哪些是资源，（实际上，有时候也不好区分，如菜单，把它理解为资源呢还是功能模块权限呢？）。其二，方便扩展，当系统要对新的东西进行权限控制时，我只需要建立一个新的关联表“权限XX关联表”，并确定这类权限的权限类型字符串。

这里要注意的是，权限表与权限菜单关联表、权限菜单关联表与菜单表都是一对一的关系。（文件、页面权限点、功能操作等同理）。也就是每添加一个菜单，就得同时往这三个表中各插入一条记录。这样，可以不需要权限菜单关联表，让权限表与菜单表直接关联，此时，须在权限表中新增一列用来保存菜单的ID，权限表通过“权限类型”和这个ID来区分是种类型下的哪条记录。

到这里，RBAC权限模型的扩展模型的完整设计图如下：

  
随着系统的日益庞大，为了方便管理，可引入角色组对角色进行分类管理，跟用户组不同，角色组不参与授权。例如：某电网系统的权限管理模块中，角色就是挂在区局下，而区局在这里可当作角色组，它不参于权限分配。另外，为方便上面各主表自身的管理与查找，可采用树型结构，如菜单树、功能树等，当然这些可不需要参于权限分配。