尚筹网

[06-后台管理系统-RBAC 权限控制模型]

1 简介

1.1 Why？为什么要进行权限控制

如果没有权限控制，系统的功能完全不设防，全部暴露在所有用户面前。用户登录以后可以使用系统中的所有功能。这是实际运行中不能接受的。

所以权限控制系统的目标就是管理用户行为，保护系统功能。

1.2 What？什么是权限控制

“权限”=“权力”+“限制”

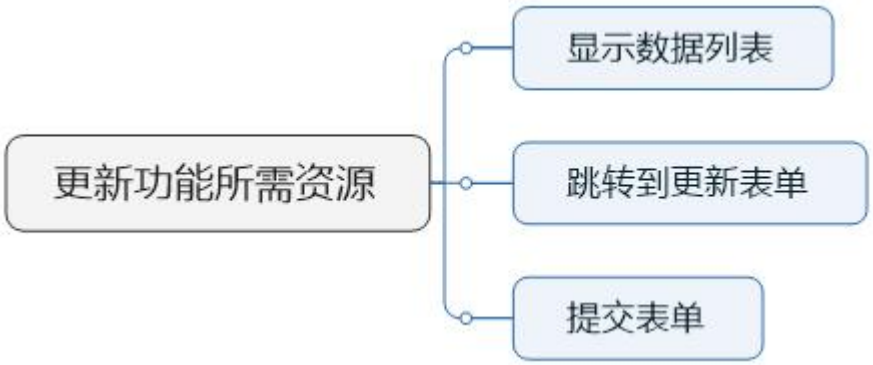
1.3 How？如何进行权限控制

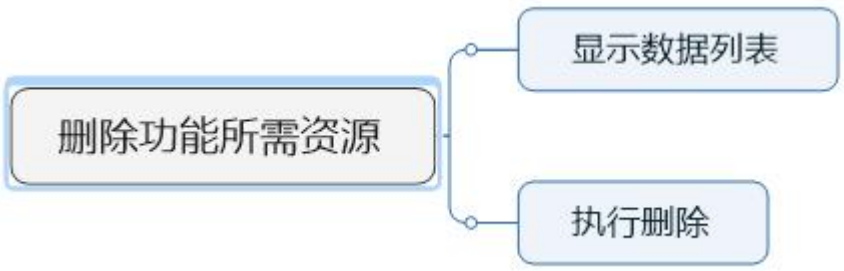
1.3.1 定义资源

资源就是系统中需要保护起来的功能。具体形式很多：URL 地址、handler 方法、service 方法、页面元素等等都可以定义为资源使用权限控制系统保护起来。

1.3.2 创建权限

一个功能复杂的项目会包含很多具体资源，成千上万都有可能。这么多资源逐个进行操作太麻烦了。为了简化操作，可以将相关的几个资源封装到一起，打包成一个“权限”同时分配给有需要的人。





1.3.3 创建角色

对于一个庞大系统来说，一方面需要保护的资源非常多，另一方面操作系统的人也非常多。把资源打包为权限是对操作的简化，同样把用户划分为不同角色也是对操作的简化。否则直接针对一个个用户进行管理就会很繁琐。

所以角色就是用户的分组、分类。先给角色分配权限，然后再把角色分配给用户，用户以这个角色的身份操作系统就享有角色对应的权限了。

1.3.4 管理用户

系统中的用户其实是人操作系统时用来登录系统的账号、密码。

1.3.5 建立关联关系

权限→资源：单向多对多

Java 类之间单向：从权限实体类可以获取到资源对象的集合，但是通过资源获取不到权限

数据库表之间多对多：

一个权限可以包含多个资源

一个资源可以被分配给多个不同权限

角色→权限：单向多对多

Java 类之间单向：从角色实体类可以获取到权限对象的集合，但是通过权限获取不到角色

数据库表之间多对多：

一个角色可以包含多个权限

一个权限可以被分配给多个不同角色

用户→角色：双向多对多

Java 类之间双向：可以通过用户获取它具备的角色，也可以看一个角色下包含哪些用户

数据库表之间：

一个角色可以包含多个用户

一个用户可以身兼数职

2 多对多关联关系在数据库中的表示

2.1 没有中间表的情况

t_subject				t_student	
id	name	fk		id	name
1	javase	1, 2, 3		1	s1
2	javaweb	1, 2, 3		2	s2
3	jdbc	1, 2, 3		3	s3
4	ssm	1, 2, 3			

如果只能在一个外键列上存储关联关系数据，那么现在这个情况无法使用 SQL 语句进行关联查询。

2.2 有中间表

t_subject			t_student	
id	name		id	name
1	javase		1	s1
2	javaweb		2	s2
3	jdbc		3	s3
4	ssm			
t_inner				
id	subject_id	student_id		
1	1	1		
2	1	2		
3	1	3		
4	2	1		
5	2	2		
6	2	3		
7	3	1		
8	3	2		
9	3	3		
10	4	1		
11	4	2		
12	4	3		

select t_studet.id,t_student.name from t_student left join t_inner on t_studen.id=t_inner.stuent_id left join t_subject on t_inner.subject_id=t_subject.id where t_subjct.id=1

2.3 中间表的主键生成方式

2.3.1 方式一：另外设置字段作为主键

t_inner		
id	subject_id	student_id
1	1	1
2	1	2
3	1	3

2.3.2 方式二：使用联合主键

t_inner			
subject_id	student_id		主键值
1	1		1, 1
1	2		1, 2
1	3		1, 3
2	1		2, 1
2	2		2, 2
2	3		2, 3

组合起来不能重复即可！

3 RBAC 权限模型

3.1 概念

鉴于权限控制的核心是用户通过角色与权限进行关联，所以前面描述的权限控制系统可以提炼为一个模型：**RBAC（Role-Based Access Control，基于角色的访问控制）**。

在 RBAC 模型中，一个用户可以对应多个角色，一个角色拥有多个权限，权限具体定义用户可以做哪些事情。

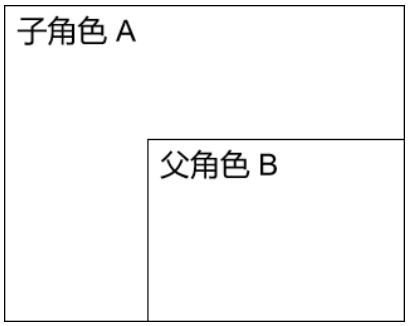
3.2 RBAC0~RBAC3

3.2.1 RBAC0

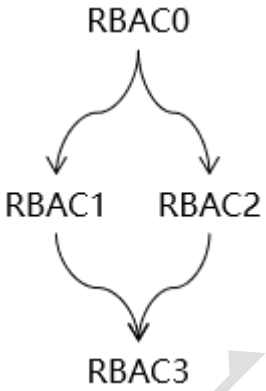
最基本的 RBAC 模型，RBAC 模型的核心部分，后面三种升级版 RBAC 模型也都是建立在 RBAC0 的基础上。

3.2.2 RBAC1

在 RBAC0 的基础上增加了角色之间的继承关系。角色 A 继承角色 B 之后将具备 B 的权限再增加自己独有的其他权限。比如：付费会员角色继承普通会员角色，那么付费会员除了普通会员的权限外还具备浏览付费内容的权限。



3.2.3 RBAC2



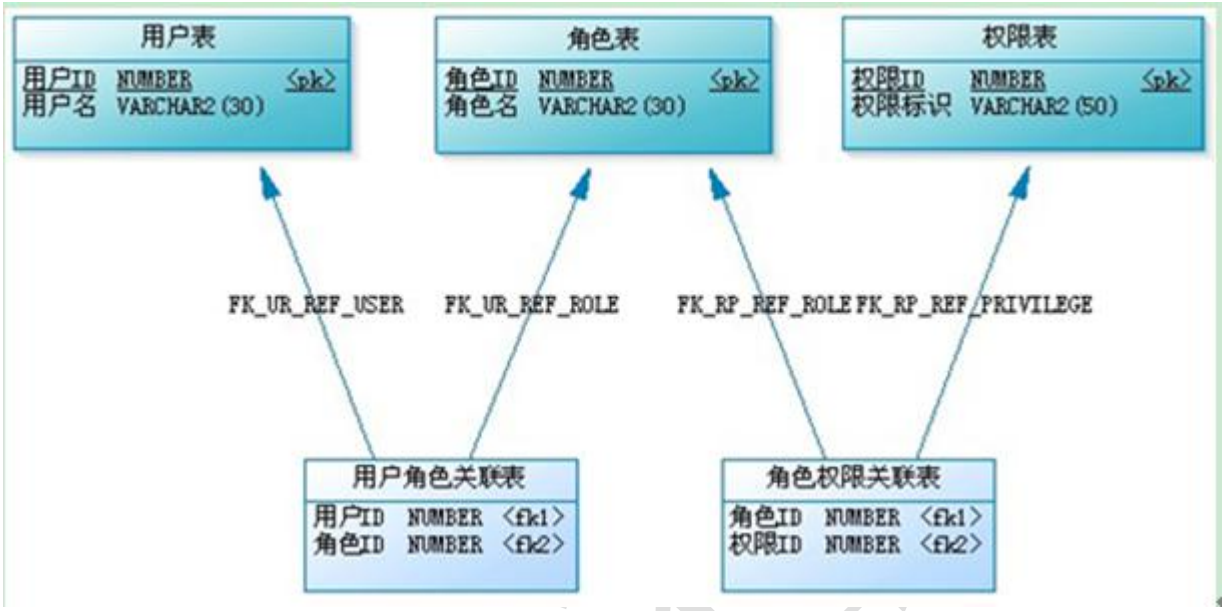
在 RBAC0 的基础上进一步增加了角色责任分离关系。责任分离关系包含静态责任分离和动态责任分离两部分。

- 静态责任分离：给用户分配角色时生效
  - 互斥角色：权限上相互制约的两个或多个角色就是互斥角色。用户只能被分配到一组互斥角色中的一个角色。
    例如：一个用户不能既有会计师角色又有审计师角色。
  - 基数约束：
    - 一个角色对应的访问权限数量应该是受限的
    - 一个角色中用户的数量应该是受限的
    - 一个用户拥有的角色数量应该是受限的
  - 先决条件角色：用户想拥有 A 角色就必须先拥有 B 角色，从而保证用户拥有 X 权限的前提是拥有 Y 权限。
    例如：“金牌会员”角色只能授予拥有“银牌会员”角色的用户，不能直接授予普通用户。
- 动态责任分离：用户登录系统时生效
  - 一个用户身兼数职，在特定场景下激活特定角色
    - 马云在阿里巴巴内部激活创始人角色
    - 马云在某企业级论坛上激活演讲嘉宾角色

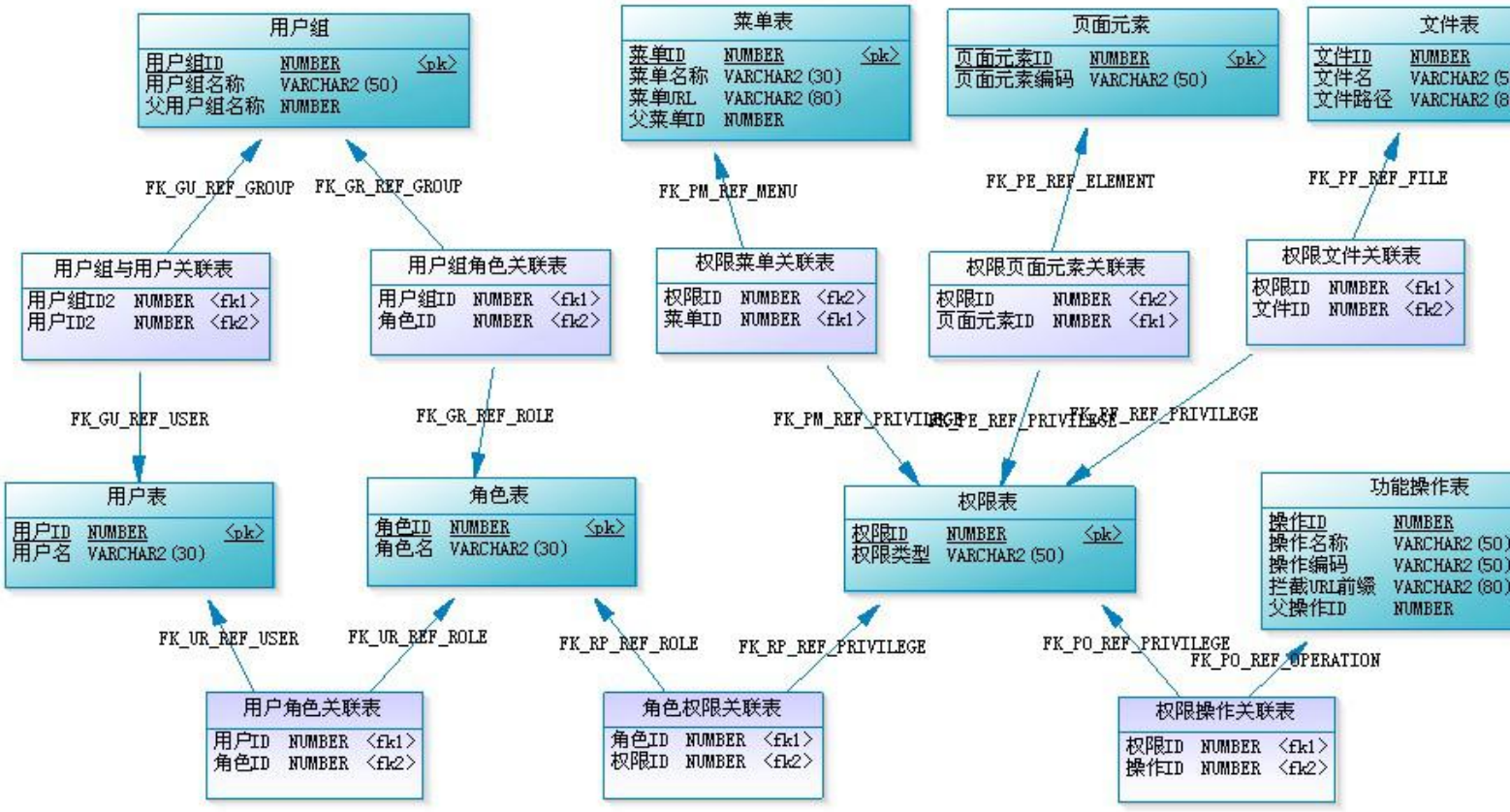
3.2.4 RBAC3

RBAC3 是在 RBAC0 的基础上同时添加 RBAC2 和 RBAC3 的约束，最全面、最复杂。

3.3 基本 RBAC 模型



3.4 扩展 RBAC 模型



（图：RBAC权限模型扩展）

3.5