资源授权系统

第一部分 理论篇

内容提要

- 权限系统研究的问题
- 常见的访问控制策略
- 基于角色的访问控制
- 诸子百家关于权限控制理论

1. 权限管理系统研究的问题

用户在什么情况下能对资源进行什么样的操作！

who（主体，用户）

what（客体，资源）

when（限制）

how（控制，操作）

1.1 主体

主体：谁准备使用、控制资源谁就是主体。

（用户，进程。。。∈主体）

描述使用资源的用户是什么样的？

--他可能处在一定的企业组织结构中，往往是一棵树

--他的权力和是组织结构相关的，他可能充当某种角色；

--主体可能是变动的，他的职位和工作可能是变化的

--给主体归类，哪些主体做一样的事情，可以认为是一类

--工作职责决定了他们的工作权力（权限）

--业务规则决定了职责访问，权限的使用条件

--用什么方式去组织我们的用户，分类我们的用户

1.2 客体

客体：客体就是一种资源。

什么是资源？只要是需要限制人使用的都是资源、

　　一个硬盘文件，url 路径

资源如何分类如何组织？

如何去描述资源？

　　--url 描述一个网页功能，文件路径文件名描述一个文件

　　资源，或者是行驶的一种权力（★）

一种基于字符串统计概率的 url 检索算法

场景描述：

用户浏览器访问

http://www.dangdang.com/refund.aspx

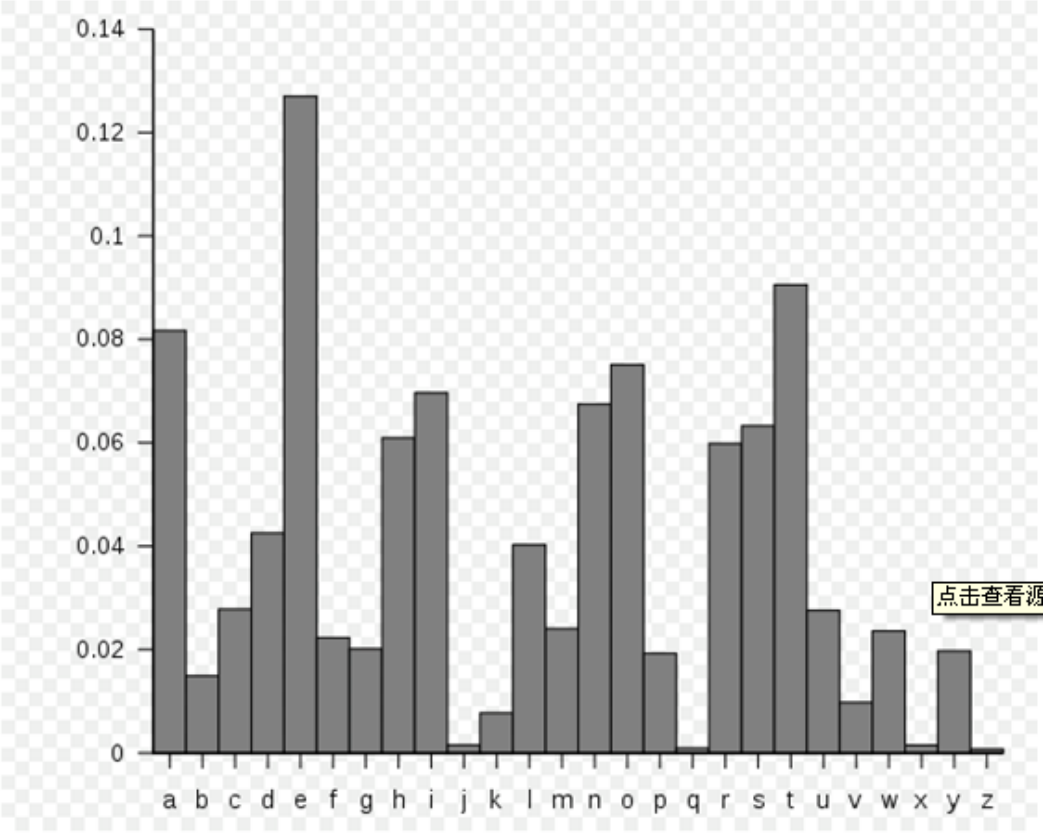
程序获取访问用户 **id** 及 **url**

http://www.dangdang.com/refund.aspx

用户号	资源编号	用户访问权
15	25545	Y
15	25546	N
15	25544	Y

资源编号	资源类型	资源描述
25544	url	http://backoffic.com/returnproduct.jsp
25545	url	http://backoffic.com/refund.jsp
25546	url	http://backoffic.com/orderdetail.jsp

字母在单词中的出现频率



字母	频率	字母	频率	字母	频率
E	0.1268	L	0.0394	P	0.0186
T	0.0978	D	0.0389	B	0.0156
A	0.0788	U	0.0280	V	0.0102
O	0.0776	C	0.0268	K	0.0060
I	0.0707	F	0.0256	X	0.0016
N	0.0706	M	0.0244	J	0.0010
S	0.0634	W	0.0214	Q	0.0009
R	0.0594	Y	0.0202	Z	0.0006
H	0.0573	G	0.0187		

取出现概率前 9 的字母做以下映射

E	T	A	O	I	N	S	R	H
1	2	3	4	5	6	7	8	0

用以上表格去映射一个 url 的有效名称，其命中率较高，如果在某一位没有命中名称取 9。

http://backoffic.com/refund.jsp

我们可以认为有效名称为 refund

命中结果为 819969

于是，得到一条记录

资源编号	资源类型	资源描述	命中数
25544	url	http://backoffic.com/returnproduct.jsp	819969
25545	url	http://backoffic.com/refund.jsp	
25546	url	http://backoffic.com/orderdetail.jsp	

当程序获取到 url 后按照同样的规则计算出命中数。根据命中数去检索 url。

原理：

1. 字符串遵循从左至右比对，比对浪费在 http://backoffic.com/的 url 前缀上
2. 用频率前 9 的字母去命中一个单词命中概率趋近于 100%

3. 命中数反应了命中的先后次序恰好命中频率前 9 的字母的组合数的概率很低，是个小概率事件。

4. 小概率事件表示事件发生性很小，而 2 保证的小概率事件发生一定能发生，从而保证通过命中数准确命中该 url 的正确性越高。

对于相同命中数问题的规避

1. 增长有效名称长度
2. 使用字母频率表的位数增加，这样命中几率进一步提高

E	T	A	O	I	N	S	R	H
1	2	3	4	5	6	7	8	0

74%命中率。

L	D	U	C	F	M	W	Y	G
1	2	3	4	5	6	7	8	0

24%命中率 合计命中>90%

3. 引入能反应 url 中字母先后次序情况的排序数
   对于没有命中的数字采用接下来的 8 个数字命中将会提高
   --- Refund 819969
   ----refund 995992
   合计命中 5/6
4. 由于字符串匹配正向匹配，url 存储倒立提高检索效率
   http://backoffice.com/refund.jsp
   倒立存储为 psj.dnufer/moc.eciffokcab//:ptth

1.3 限制（规则）

限制：说明在什么情况下有权利使用资源。

--他是一种访问控制规则，明确了一类人对资源的使用权利。
--可能是基于一种业务规则，比如会计和出纳；
--他和正在实施的任务有关
--他可能是一种程序或者人为的限定，比如私有权利不被继承。

1.4 控制

控制：对资源具体的操作
--告诉主体能够对资源有哪些操作
--告诉主体对某个资源是否能够具有某种权利的操作
--权限管理系统不负责你对具体资源的实施
--权限管理系统只能告诉对客体有没有某种操作权限，没有办法控制你具体的操作
--鉴权或者控权系统则负责具体行为的限制

2. 常见的访问控制策略

2.1 自主访问控制

用户（主体）自己决定对资源的访问权限
用户对资源的访问权作为用户的一种属性（特性）
用户可以把自己的权限授予子用户—授权自主
这是一个从用户到资源的直接关系。

	资源 1	资源 2	资源 3	资源 4
用户 1	W/R	W/R	R	R
用户 2	W	W	W	W
用户 3	W/R	W/R	W/R	W/R
用户 4	W/R/X	W	W	W

优点：灵活

缺点：不适合统一管理，不适合人员众多重复变动场合。

2.2 强制访问控制

系统首先预定义一套访问策略，信息分密级和类进行管理，以保证每个用户只能访问到那些被标明可以由他访问的信息的一种访问约束机制

主体和客体都提前定义了安全属性

用户不能轻易改动这套访问策略

当用户本身的安全属性和客体能允许访问的安全属性匹配时则能够访问

如 windows 中超级管理员，来宾账号，管理员这套策略

优点：

安全性高，

管理集中

缺点：

不够灵活

不够灵活，

访问策略是提前预定义好了的，不适用于不断变化的系统环境

2.3 基于角色的访问控制

（RBAC，RoleBasedAccessControl）

NIST（The National Institute of Standards and Technology，美国国家标准与技术研究院）

分配给用户一个角色，而不是给每个用户单独分配权限；
角色反应的是一类用户所有权限的共性，可以根据业务需要提前设定；
适合用户的变动环境，授权管理成本下降；
更容易和实际业务关系结合起来。

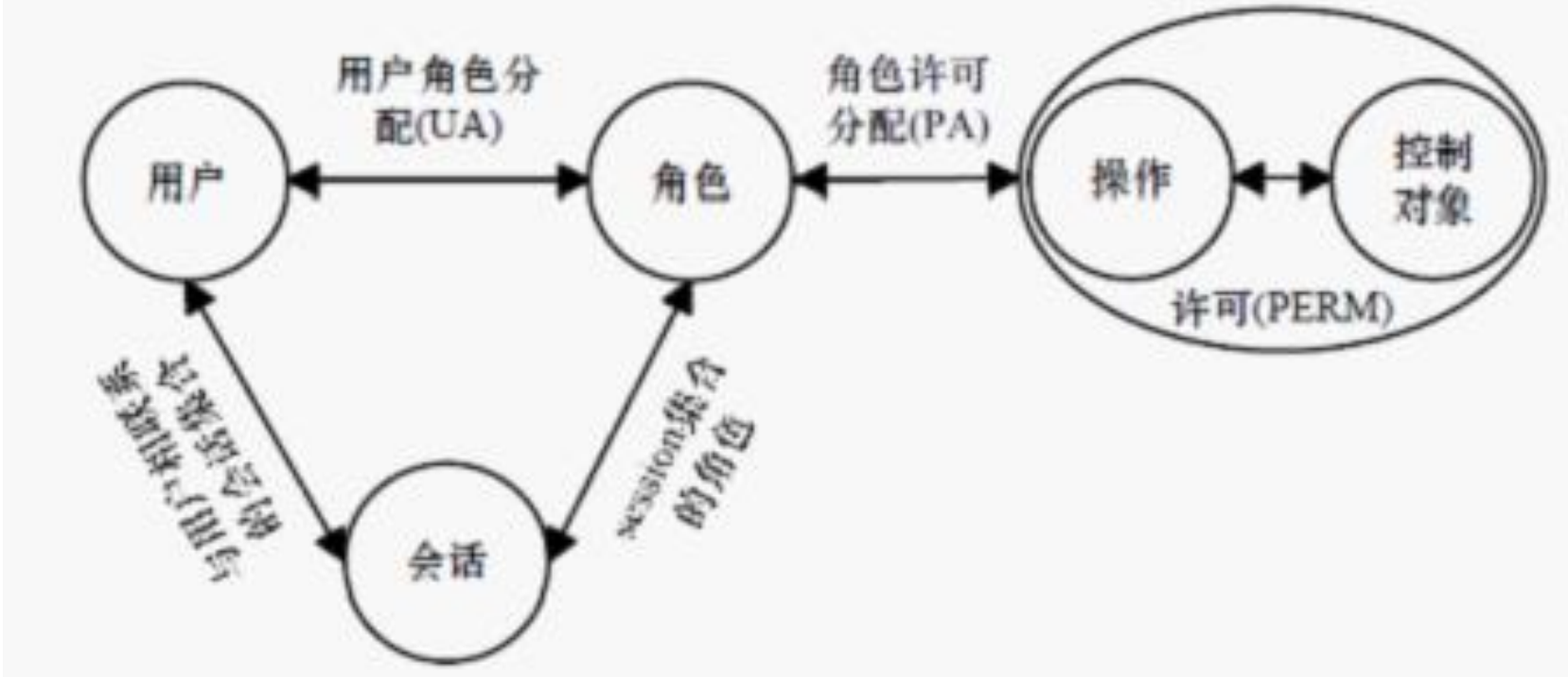
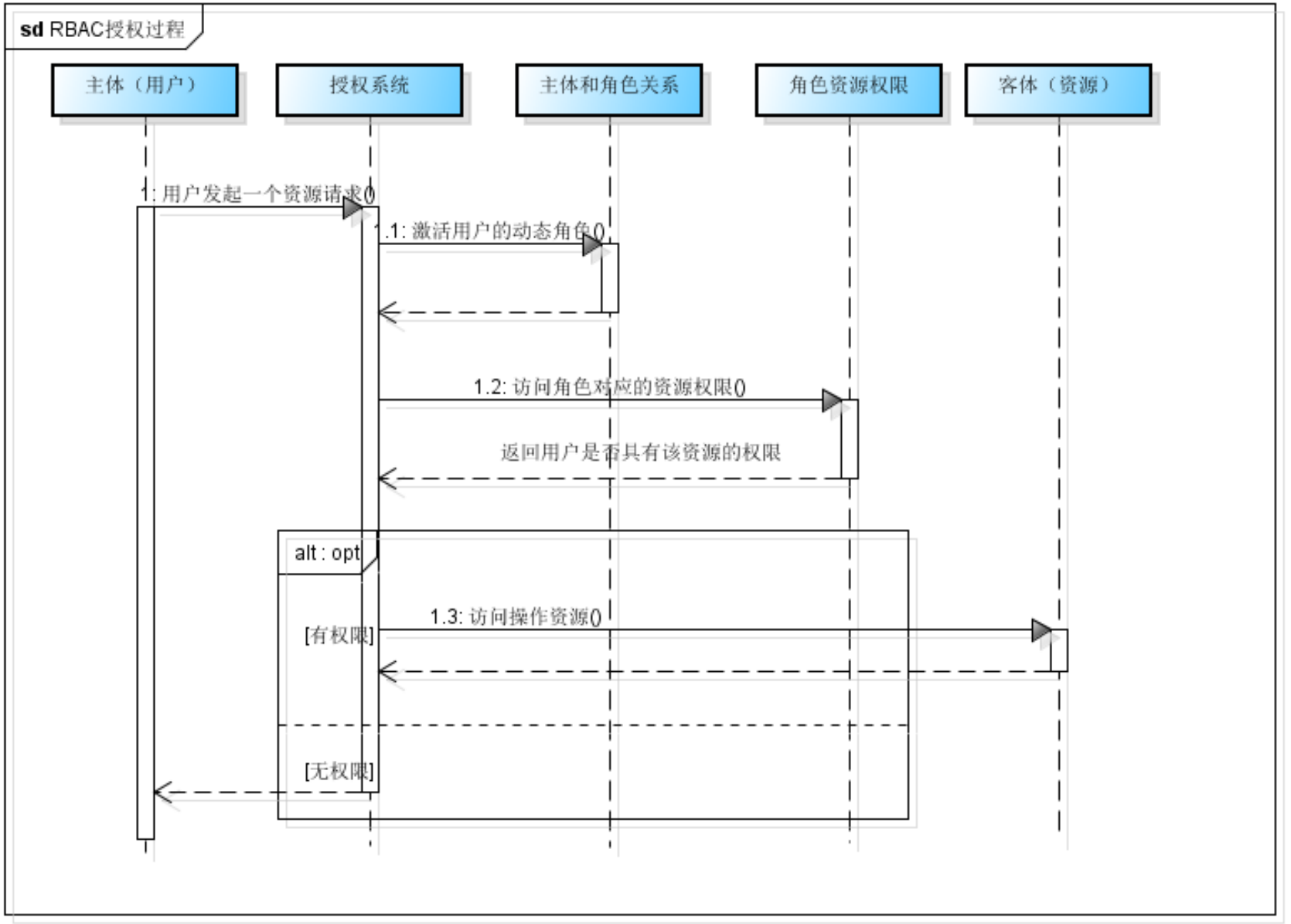


图 1　Core RBAC 模型


sd RBAC授权过程
主体（用户）
授权系统
主体和角色关系
角色资源权限
客体（资源）
1: 用户发起一个资源请求()
1.1: 激活用户的动态角色()
1.2: 访问角色对应的资源权限()
返回用户是否具有该资源的权限
alt : opt
[有权限]
1.3: 访问操作资源()
[无权限]

RBAC 名称概念

(1)主体

谁想享有资源，操作资源谁就是主体，主体可以是人等

(2)用户

主体的一部分，通常情况下我们可以认为是主体

(3)用户组

对用户的一个组织结构划分，权限不考虑分配给特定的用户而给组。组可以包括组(以实现权限的继承)，也可以包含用户，组内用户继承组的权限

(4)组织结构

更为贴近企业组织关系的结构，通常是一个树形结构

(5)会话

一次访问/验证某个主体是否对某个资源具有某种权限的过程叫会话，或者和系统的一次交互过程，一个用户可以同时开启几个会话，一个会话又可以在不同情况下激活多个角色（如继承了多个角色的用户），应该根据实际的规则限制激活会话

(6)客体

资源，主体要访问资源，有很多中形式，资源具有层次关系和包含关系，例如，网页是资源，网页上的按钮、文本框等对象也是资源，是网页节点的子节点，如可以访问按钮，则必须能够访问页面。

资源描述，大体来说可以是纳入系统管理的信息，在技术实现层面可以是一张表、一条或一列记录、甚至可以是表的一个单元格。
资源分类，不同的资源描述特性不同,需要分类处理.
资源权限集合,资源本身没有权限的这种概念，资源的这种权限的概念是由使用产生的

（7）权限
表示对一个客体（资源的访问控制标识）对受保护的资源操作的访问许可,是绑定在特定的资源实例上的
权限粒度,把权限分离到多大合适,
权限集合,该资源所拥有的所有的权限集合

用户的权限计算公式

用户权限==所在组权限+角色权限+其继承的权限

A．权限的硬编码实现

资源号	7	6	5	4	3	2	1	0
权　限	置顶	评论	回复	转载	分享	删除	编辑	查看
1009	1	1	1	1	1	1	1	1
1010	1	1	0	0	0	1	1	0

预定 **byte** 字节中每一位的权限含义,如果有该位权限则为 **1**

若用户 **A** 对资源 **1010**,拥有查看,编辑,分享权限则有:

A 的权限为　00000001　　1

00000010　　2

00001000　　8

00001011　　11

计算机存储形式为

序号	权限描述	权限值
1	查看	1
2	编辑	2
3	删除	4
4	分享	8
....	...	...

于是对 A 授予一种权限则为 （+）

P(A)=P(A)|p

对 A 授予查看,编辑,分享权限为,

P(A)=0 | 1 | 2 | 8=11

解除 A 的 查看 权限（-）

P(A)=P(A)&(~(p))

P(A)=P(A)&(~(1))

判断 A 是否有 编辑 权限

P(A)&p==p?

P(A)&2==2?

特点：

权限的最大值由选用的数据类型决定，选用 int 可存 32，long 可存 64，权限范围有限

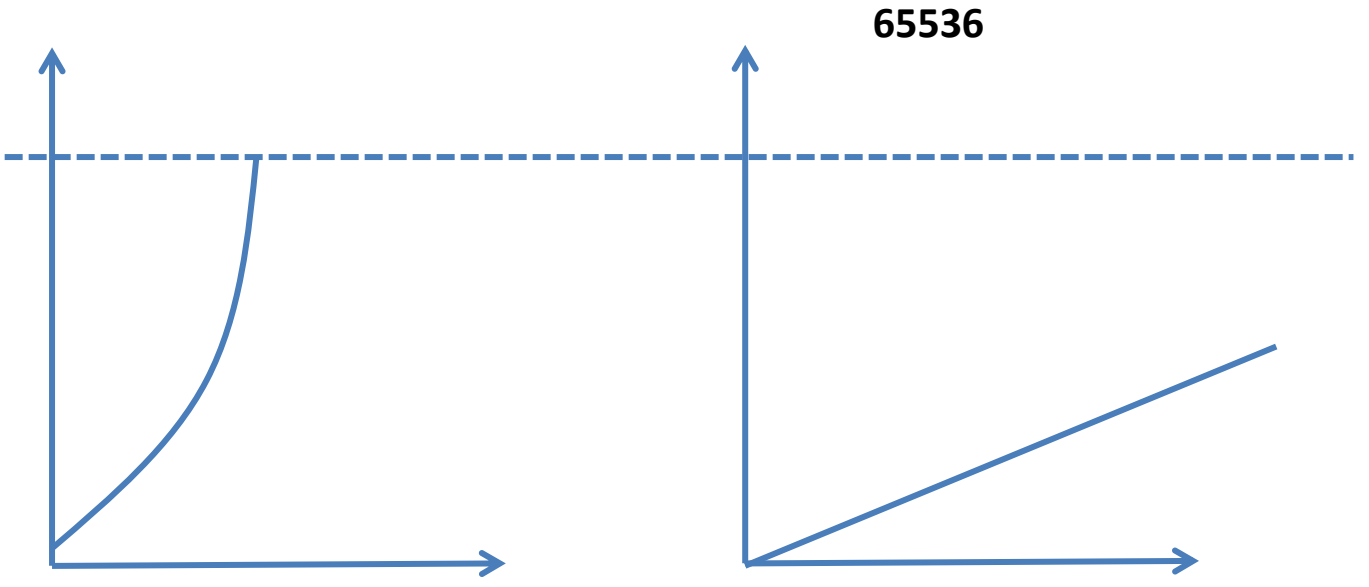
容易理解，计算简单，扩充性问题

B．基于素数特性的编码

权限描述	权限值	最大生成权限范围
查看	1	1
编辑	3	1, 3, 4
删除	5	1, 3, 4, 5, 6, 8, 9
分享	11	1, 3, 4, 5, 6, 8, 9，11, 12, 14, 15, 16, 17, 19, 20
…	21	。。。

特点：

虽然压缩性提高表示的权限范围提高，不容易理解，是一种很好的信息压缩算法。



C 实际工程应用

实际工程应用中可能不会直接采取以上两种做法，直接用数据库的存储取代这种算法的计算。

资源号	*权限描述*	权限值
1000	*查看*	1
1000	*编辑*	2
1000	*删除*	4
1000	*分享*	8
1001	*查看*	1
1001	*编辑*	2
….	…	…

最小权限原则：
有选择的将权限赋予用户
被赋予的仅仅是完成工作必需要的权限，避免越权的威胁

(8)角色

角色是用户在某个环境中的身份，这个身份拥有某些相匹配的权限，是一个授权的集合。
他表示了对若干资源的访问特性，可能和组织结构中的职称吻合，但不是等同于。
一个用户可以有多个角色，用户所具有的权限是由用户拥有的角色权限集合组成，而这些权限是相对固定

3. 基于角色的访问控制

(1) RBAC0

构成 rbac 的最基本需要，是构成权限控制的最小集合

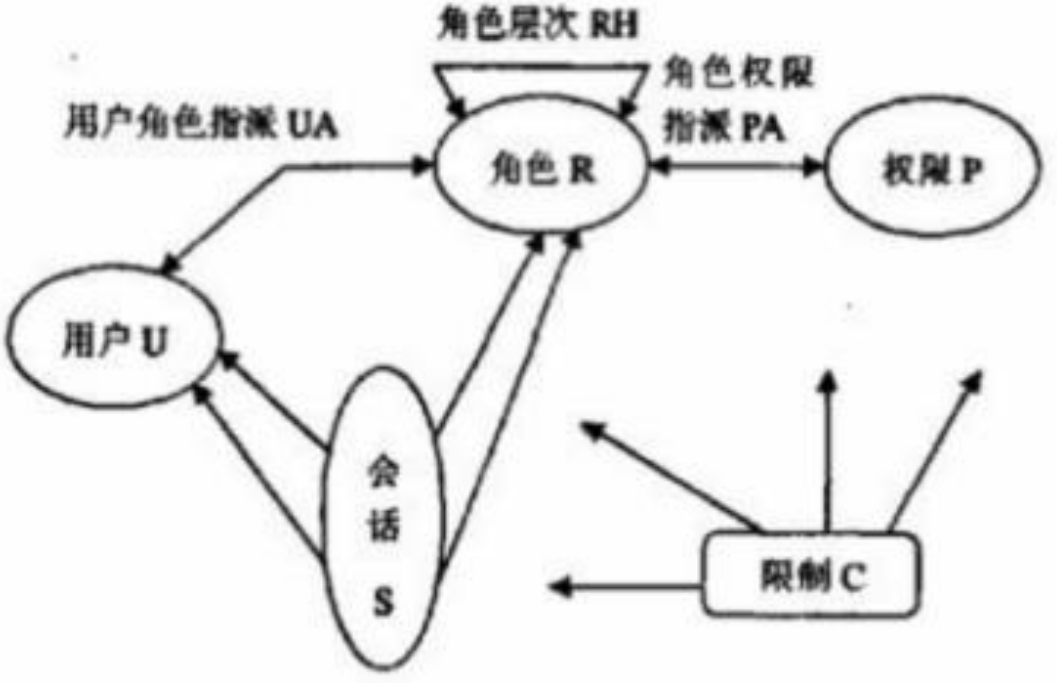


图 2.2 RBAC96 模型

在 RBAC 之中,包含用户 users(USERS)、角色 roles(ROLES)、目标 objects(OBS) 、 操 作 operations(OPS) 、 许 可 权 permissions(PRMS)五个基本数据元素，**权限被赋予角色,而不是用户，当一个角色被指定给一个用户时，此用户就拥有了该角色所包含的权限**。

RBAC0 与传统访问控制的差别在于增加一层间接性带来了灵活性，RBAC1、RBAC2、RBAC3 都是先后在 RBAC0 上的扩展

限制 ：是人为设定的需要遵守的规则(*前文已描述*)，用户不应该超过其角色限定的权限范围或者超越职责行驶权力

(2) RBAC1

引入了角色继承的概念

角色不是孤立存在的，而是在一定的环境中；

有些权限是可能存在复用的；

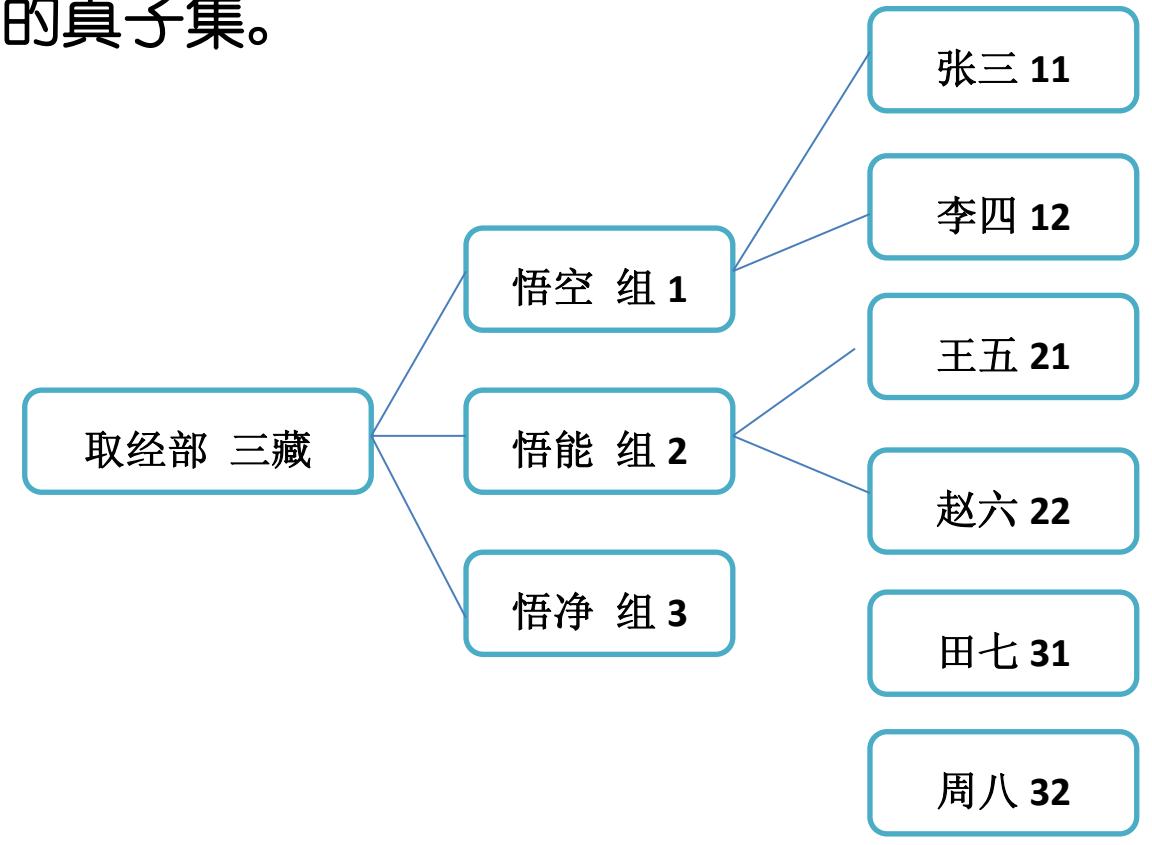
角色和实际的组织关系职责大小和实际业务要结合。

角色继承

RBAC1 引入角色间的继承关系，角色间的继承关系可分为一般继承关系和受限继承关系。一般继承关系仅要求角色继承关系是一个**绝对偏序关系**，允许角色间的多继承。而受限继承关系则进一步要求角色继承关系是一个**树结构**。

绝对偏序关系 <=

集合 A（1,3,5,6,7）若 B 继承与 A ，则 B 的取值是最大是 A 的真子集。



虚角色

定义虚角色为了体现很多角色的共性和层次性

不分配给实际的用户，而是作为一种共性被其他角色继承
如我们可以为一个组定义一个虚角色，在一个组的用户自然具有了组的权限。

实角色
略。

(3) RBAC2

RBAC2 模型中添加了责任分离关系。

职责分离,

现实领域,为了安全起见,用户不允许独立完成某个操作,而是由不同用户去完成,防止权力滥用。

角色互斥

权限互斥，对于任意两个访问权限 p1 和 p2，任何用户都不能同时拥有它们，则 p1 和 p2 互斥。

若 p1，p2 分别分配给你 r1，r2 则角色 r1 和 r2 互斥。

银行系统中，如果某个用户是贷款角色中的一员，那么这个用户绝对不可以再属于借款角色
责任分离包括静态责任分离和动态责任分离。
静态职责分离
互斥的两个角色不应该赋予同一个用户
如会计出纳,先天存在职责分离

动态职责分离
用户可以同时拥有互斥的角色,但是每次只能行驶一个角色的权力.

案例：公文流转

----（1）起草
----（2）审批
----（3）发布
公文起草者和审批不能分配给同一个用户，静态职责分离
公文的起草和发布可以是同一个用户，但是同时只能激活一个角色

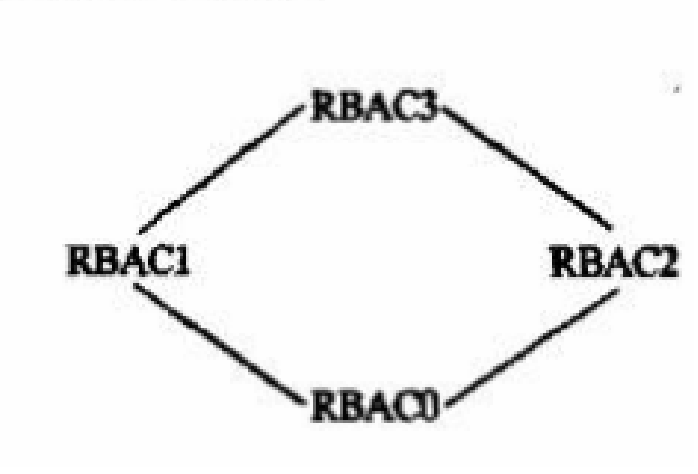
静态职责分离在给角色授权的时候就已经存在的现状。
动态职责分离是在角色激活的时候作用，他需要对

角色授权方式
自主访问控制.向下授权过程

基于角色访问控制,权限是一个累积的过程

(4) **RBAC3**

RBAC3 包含了 RBAC1 和 RBAC2, 既提供了角色间的继承关系, 又提供了责任分离关系。

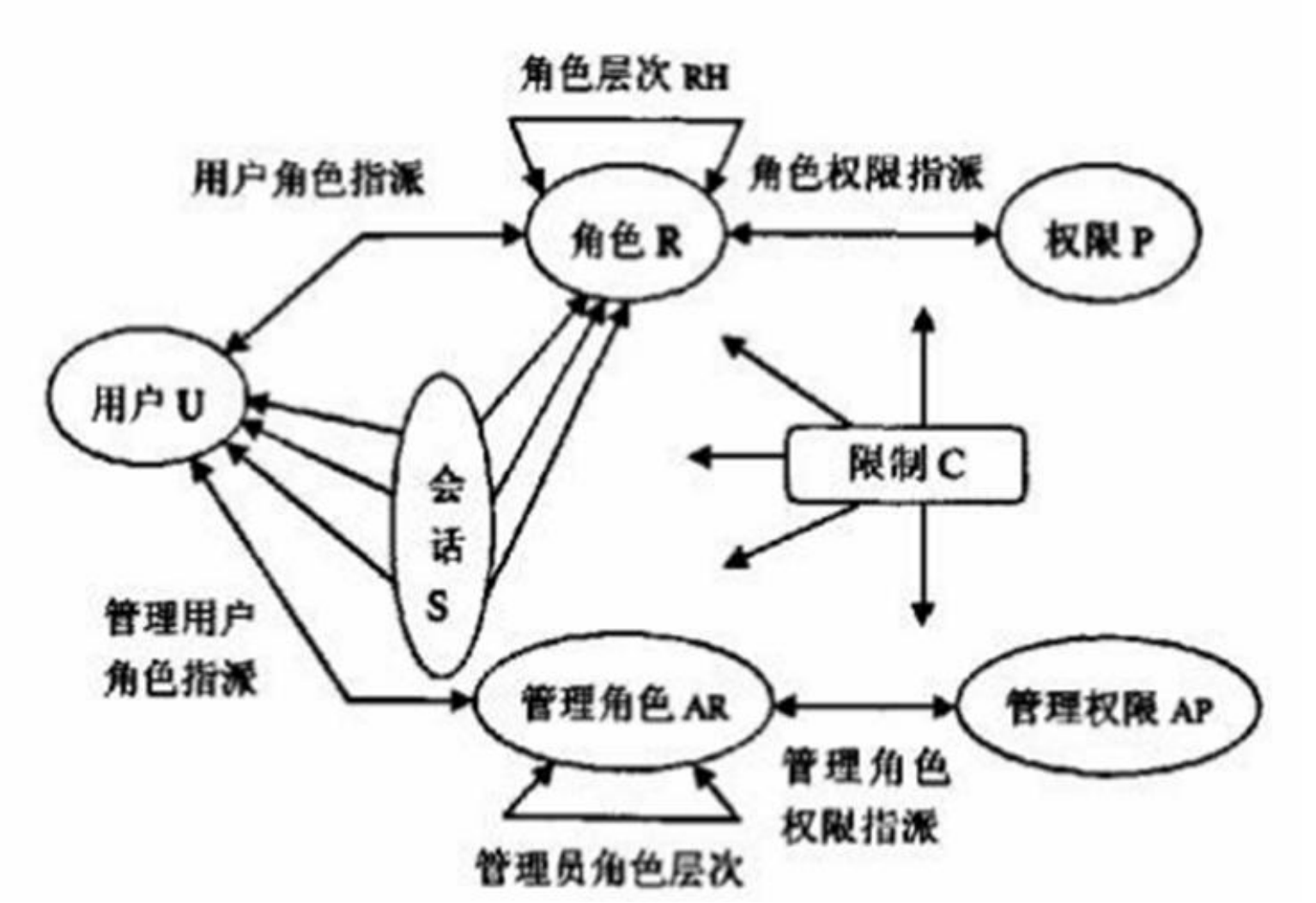


RBAC2 并不是 RBAC1 的升级。

RBAC96 权限配置过程

a. 系统管理员根据系统功能定义权限
b. 根据组织关系或任务中的工作职责创建角色
c. 根据限制，定义角色继承关系
d. 将权限赋予角色
e. 激活角色
f. 权限判定
g. 操作资源

RBAC97


角色层次 RH
用户角色指派
角色权限指派
角色 R
权限 P
用户 U
会话 S
限制 C
管理用户
角色指派
管理角色 AR
管理权限 AP
管理角色
权限指派
管理员角色层次

所有的权限管理工作由一个系统管理员配置“鸭梨山大”
一个管理员不可能对系统所有的情况了解。

建立一个管理角色，负责角色权限配置，他拥有管理权限。
可以定制多个区域管理员对系统分布/分类管理，每个管理
员负责一个区域/类别的管理，权限互不交叉。

不管是 RBAC96/97 管理员只有管理权限系统的权利，而不
是具有系统所有资源的访问权。

4. 关于权限管理的诸子百家理论

4.1 角色生命周期概念

一个角色的使用是有期限限制的,不可能永远有效;
在特定的情况下才能启用特定角色;

4.2 基于任务的访问控制

理由:
角色不是永远有效，可能只在一个有限的时间内，一定的工作任务下有效
同一个角色在不同的情景下的权限是会发生变化的

以任务项目为中心
如项目经理当项目结束之后他的职权就消失了

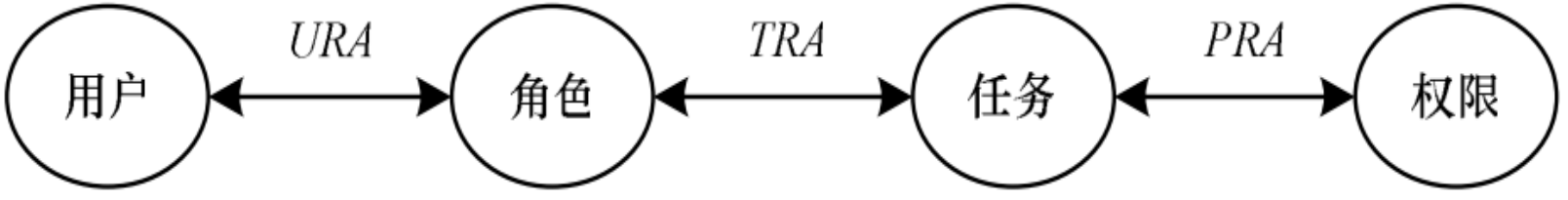
实际工作中我们总是把任务分成工作流，也就是业务流程，在不同的环节完成工作的用户发生变化，角色也就会发生变化，用户的权限也会变化传统的 rbac 解决不了这种具有工作流性质的问题

一个合适的访问控制机制应该保证授权只有在任务开始执行时才授予且任务结束授权就要被收回，否则可能导致安全泄露.

定义

基于任务的访问控制模型 就是面向上下文环境的访问控制。

(a)RBAC 模型 3 层结构



(b)T-RBAC 模型 4 层结构

一种动态授权的主动安全模型，主体仅在执行任务时访问客体，任务结束权限就被消耗完，防止主体对客体权限的无限期拥有，同时客体的访问控制权限随着执行任务的上下文环境变化而变化

组织：
用户不是孤立的是有一定联系的，企业的组织结构，一定程度上反应了用户的权限等级；

工作岗位：
工作岗位和角色是很相似的，但是有区别，工作岗位强调的是企业管理，而角色强调的是工作内容，一个岗位不仅仅只

做一个工作内容，（兼职）两种不可等同，但是用户的工作岗位为用户角色建模提供很好的参考意义，但是我们不能忽略其偏差

业务规则：企业的业务规则决定了角色的职责分离
业务流程：业务流程的变换决定了用户的变化，权限的变化

动态授权
建立一个权限变更队列
　　一个角色执行完一个任务之后自动取消当前的权限，然后赋予下一个身份的权限

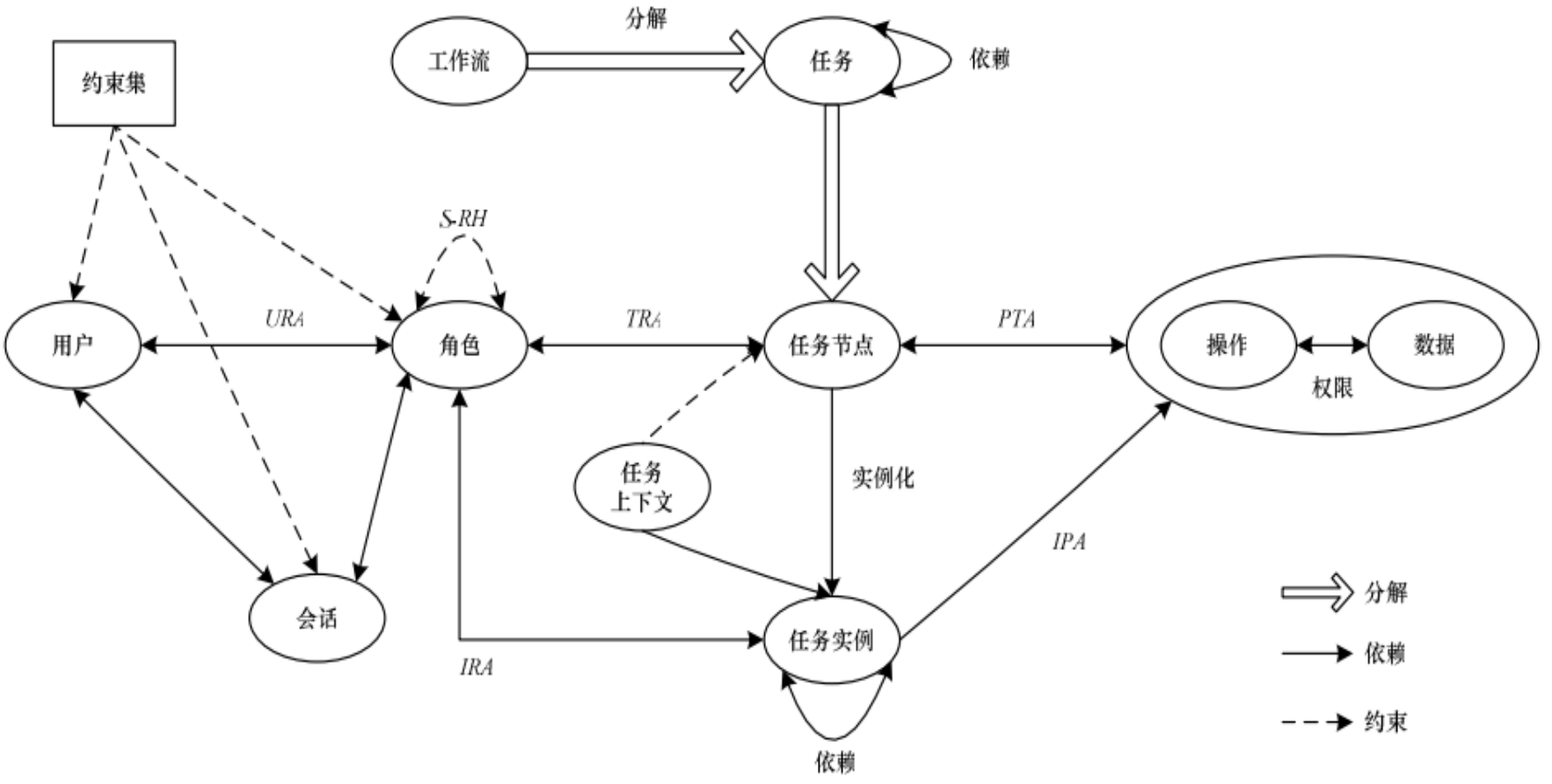
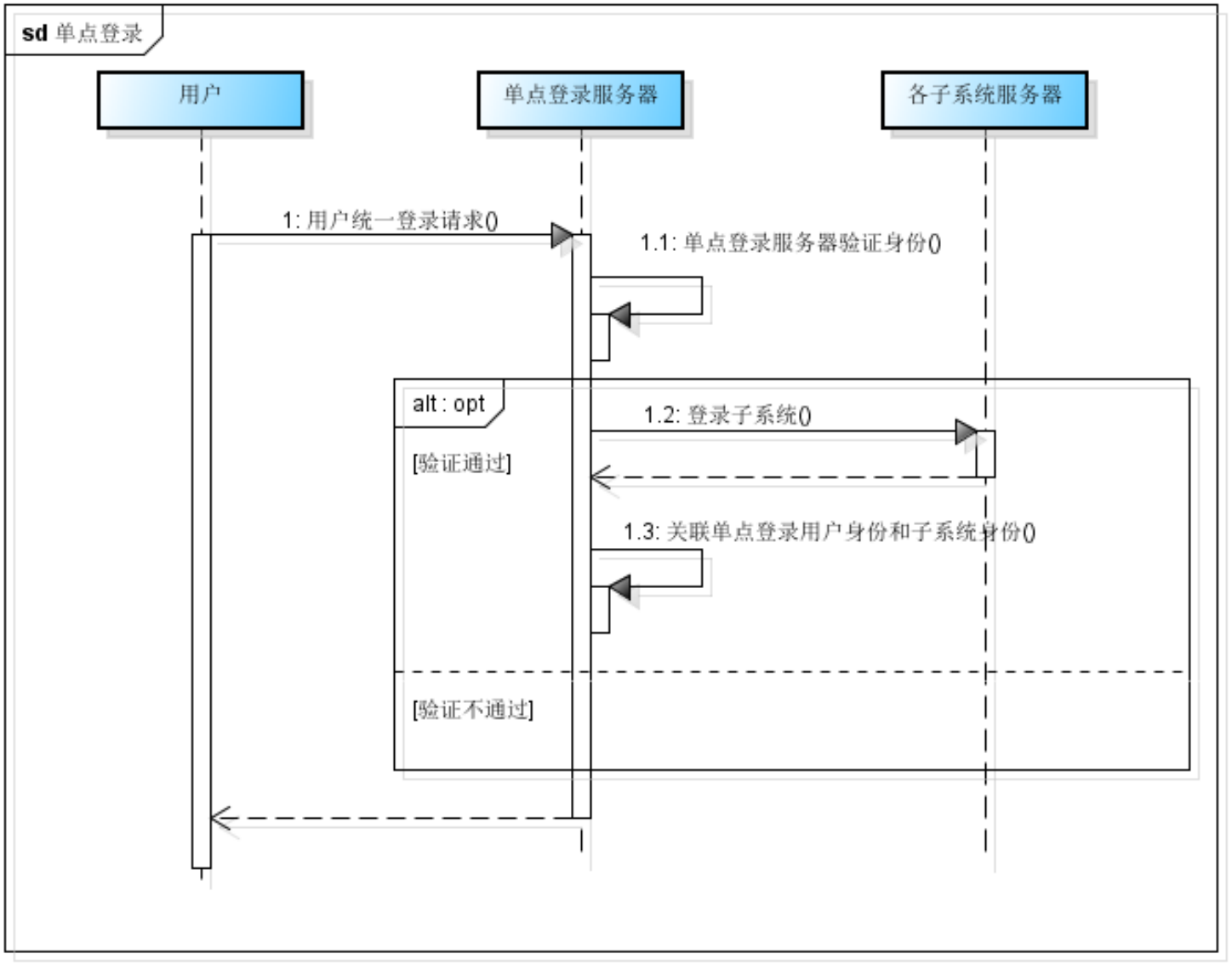
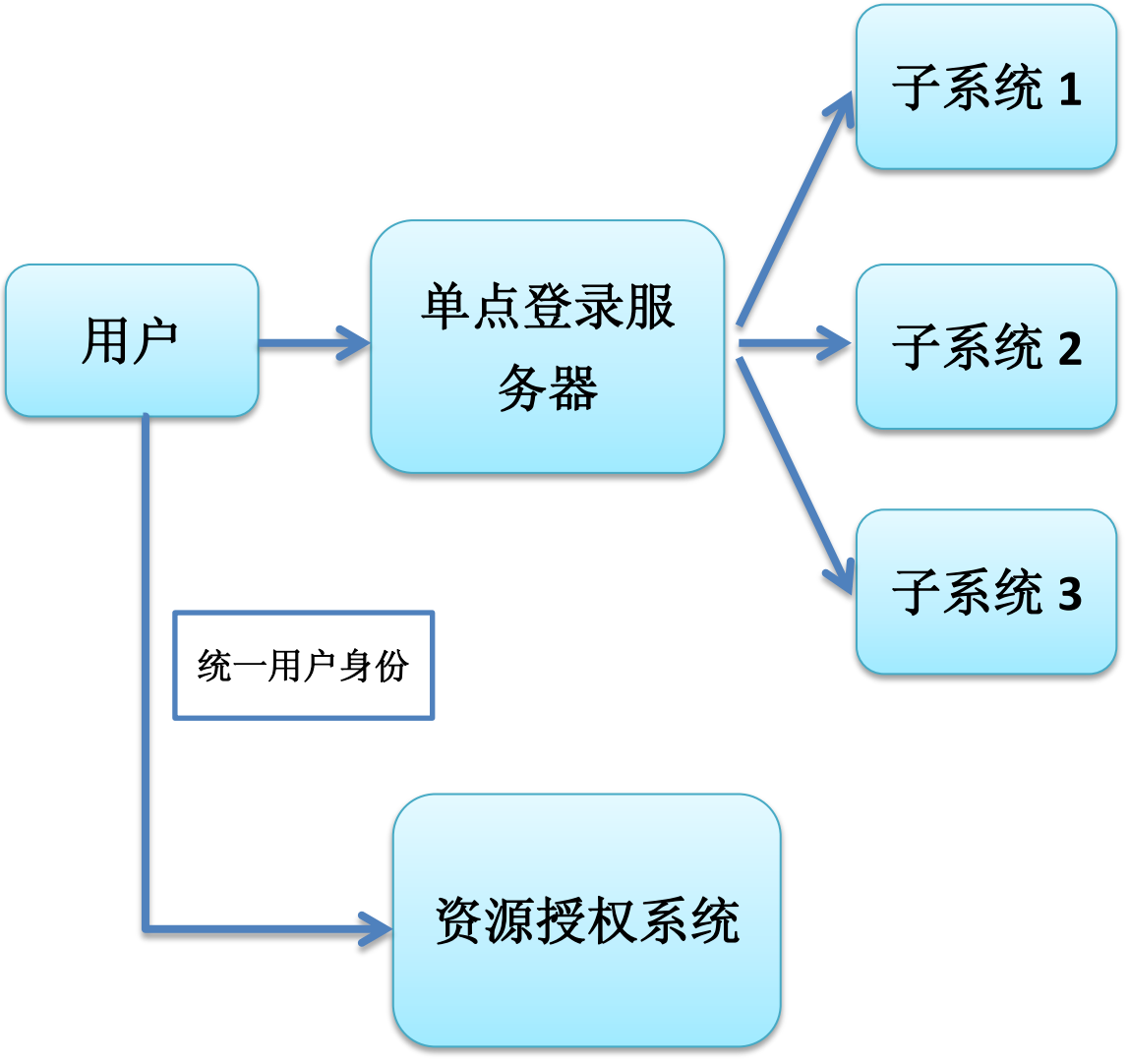

约束集
工作流
分解
任务
依赖
S-RH
用户
URA
角色
TRA
任务节点
PTA
操作
数据
权限
任务
上下文
实例化
IPA
会话
IRA
任务实例
依赖
分解
依赖
约束

图 2　T-RBAC 模型

4.3 单点登录，多点通行

一个企业可能有多个系统，多个系统都有自己的一套登录入口，如果我们进入一个系统都要重新登录将会十分麻烦，不利于权限系统的整合，控制系统要满足单点登录
采用单点登录之后（单点登录如何实现）


sd 单点登录
用户
单点登录服务器
各子系统服务器
1: 用户统一登录请求()
1.1: 单点登录服务器验证身份()
alt : opt
[验证通过]
1.2: 登录子系统()
1.3: 关联单点登录用户身份和子系统身份()
[验证不通过]


用户
单点登录服务器
子系统 1
子系统 2
子系统 3
统一用户身份
资源授权系统

4.4 监管继承，工作职责的代理

缺席角色（关键角色）

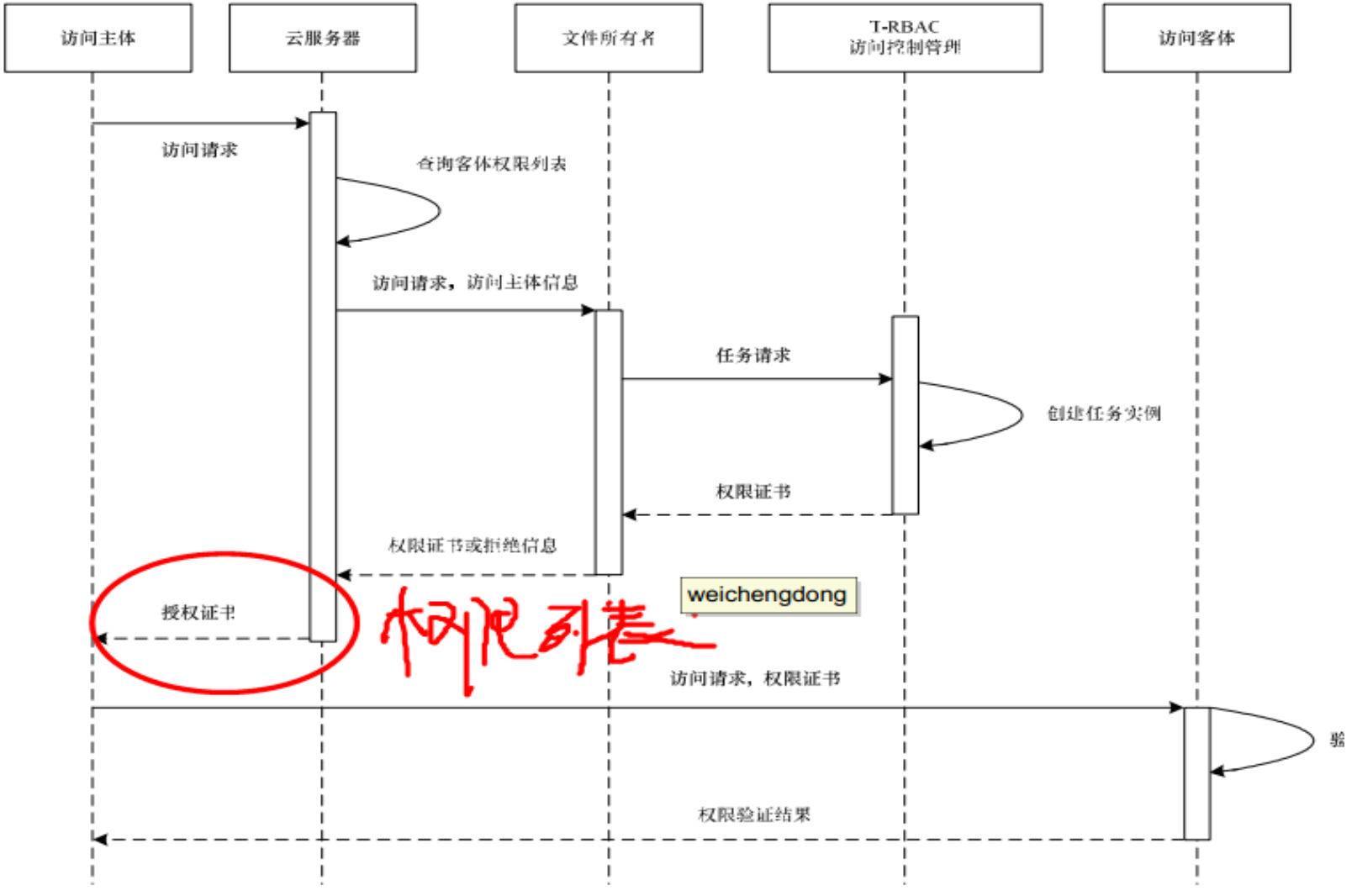
并不是任何条件下都可以继承，只有满足一定条件是才能继承权限，如只有某人请假时才能把他的权限授予他人执行工作。

监管继承应保证数据的完整性和一致性。

4.5 泛化继承

根据业务具体的抽象关系决定角色的继承关系

4.6 一种基于任务的云计算访问模型、


访问主体
云服务器
文件所有者
T-RBAC
访问控制管理
访问客体
访问请求
查询客体权限列表
访问请求，访问主体信息
任务请求
创建任务实例
权限证书
权限证书或拒绝信息
授权证书
weichengdong
访问请求，权限证书
权限验证结果

4.7 角色继承的面向对象概念提出

表 1　角色继承机制

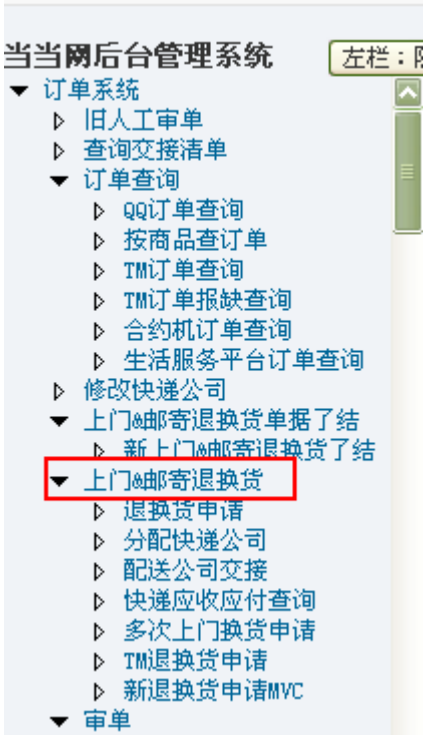
Tab.1　The role-inherited mechanisms

权限性质	继承方式	能否继承	继承后权限性质
公有权限(PBP)	公有继承	能	公有权限(PBP)
受保护权限(PTP)		能	受保护权限(PTP)
私有权限(PRP)		否	—
公有权限(PBP)	受保护继承	能	受保护权限(PTP)
受保护权限(PTP)		能	受保护权限(PTP)
私有权限(PRP)		否	—
公有权限(PBP)	私有继承	能	私有权限(PRP)
受保护权限(PTP)		否	—
私有权限(PRP)		否	—

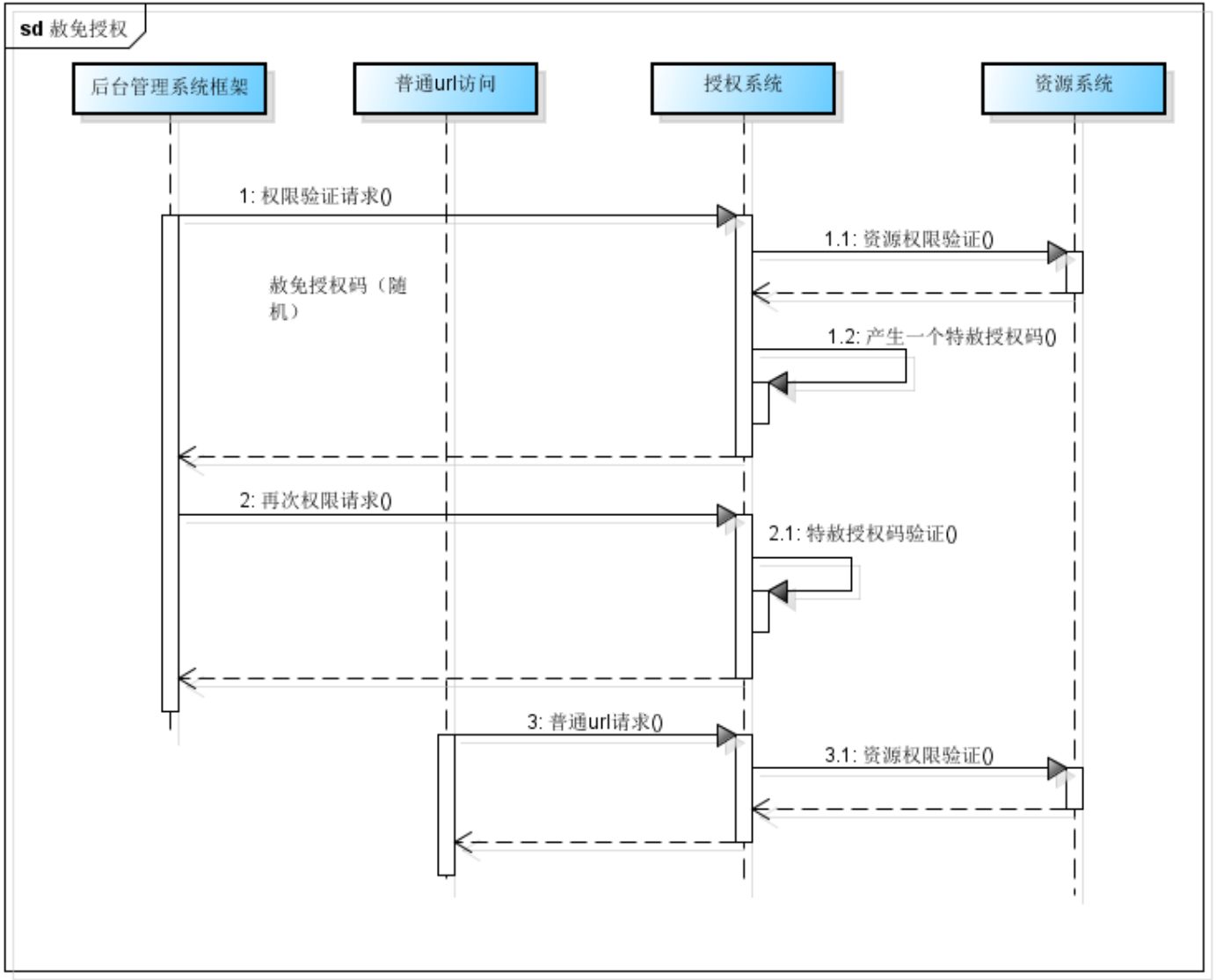
公有权限，保护权限，私有权限
权限重载
当角色自己的权限和其继承的角色重载时采取的策略

4.8 登录验证与实时验证

4.9 节点授权


当当网后台管理系统
左栏：
订单系统
旧人工审单
查询交接清单
订单查询
QQ订单查询
按商品查订单
TM订单查询
TM订单报缺查询
合约机订单查询
生活服务平台订单查询
修改快递公司
上门&邮寄退换货单据了结
新上门&邮寄退换货了结
上门&邮寄退换货
退换货申请
分配快递公司
配送公司交接
快递应收应付查询
多次上门换货申请
TM退换货申请
新退换货申请MVC
审单

4.10 授权赦免


sd 赦免授权
后台管理系统框架
普通url访问
授权系统
资源系统
1: 权限验证请求()
1.1: 资源权限验证()
赦免授权码（随机）
1.2: 产生一个特赦授权码()
2: 再次权限请求()
2.1: 特赦授权码验证()
3: 普通url请求()
3.1: 资源权限验证()

4.11 统计授权

最近最频使用

4.12 授权缓存队列,

在一定时间授权过的资源暂时放在一个缓存队列

4.13 关联授权

（操作系统页式存储，段页式存储）

参考文献

[1]. 孔广黔，李坚石,郭晓明基于 RBAC 的职责分离约束关系研究[N]

[2]. 李健，陈杰，RBAC 模型权限管理中三种新的角色继承机制和授权策略[J]

[3]. 张世龙，沈玉利,RBAC 模型中角色继承关系的研究与改进[J]

[4]. 李兰崇,基于角色的权限管理访问控制系统平台研究与实践[N]

[5].王小威，赵一鸣一种基于任务角色的云计算访问控制模型一种基于任务角色的云计算访问控制模型[J]