

RBAC的设计与具体实现

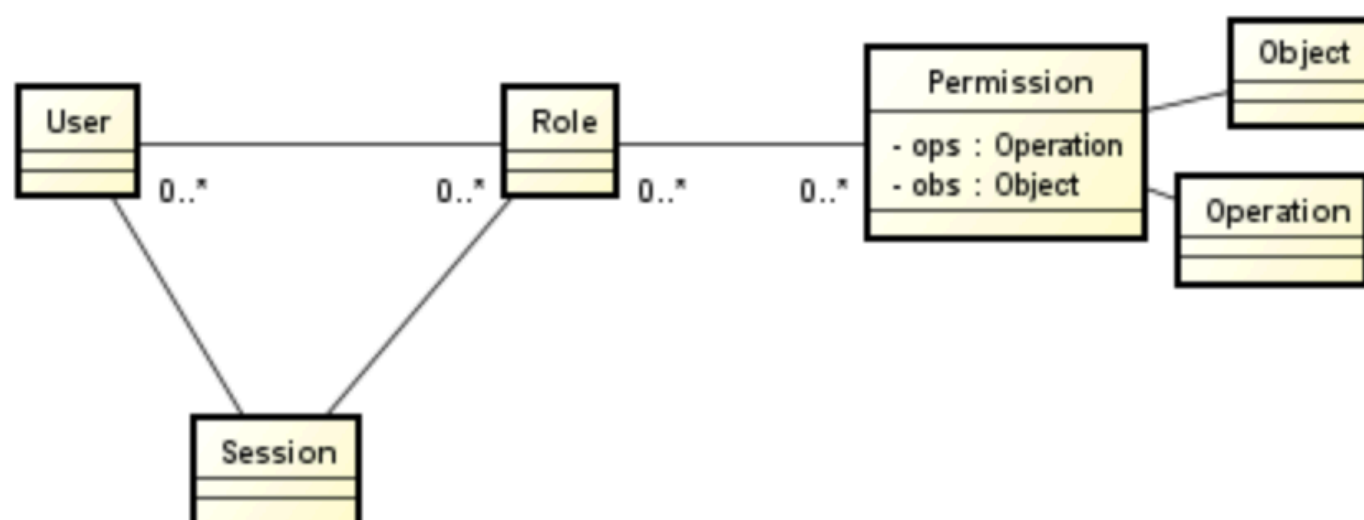
简介

RBAC (role-based access control) 是一种最基本的也是最重要的系统组成部分。RBAC 系统顾名思义是基于角色的访问控制权限系统，所有具有权限管理功能的系统都应该含有 RBAC 系统。它可以做到根据不同的用户，控制不同的用户角色，而不同的用户角色又有不同的权限。权限则表示着可以对控制对象有不同的操作。其设计的关键部分就在于如何控制角色，用户和权限的关系，下面开始对 RBAC 系统做具体的分析。

具体分析

RBAC 基本角色包含四类：用户（User），角色（Role），权限（Permission），以及会话（Session）。应该实现的基本功能有：用户的角色分配，角色的权限分配，以及一个用户在不同场景下不同角色的分配。此外，也可以加上对用户的鉴权功能，即鉴定这个操作发生发生时，是否有用户登录和登录用户是否属于该系统。

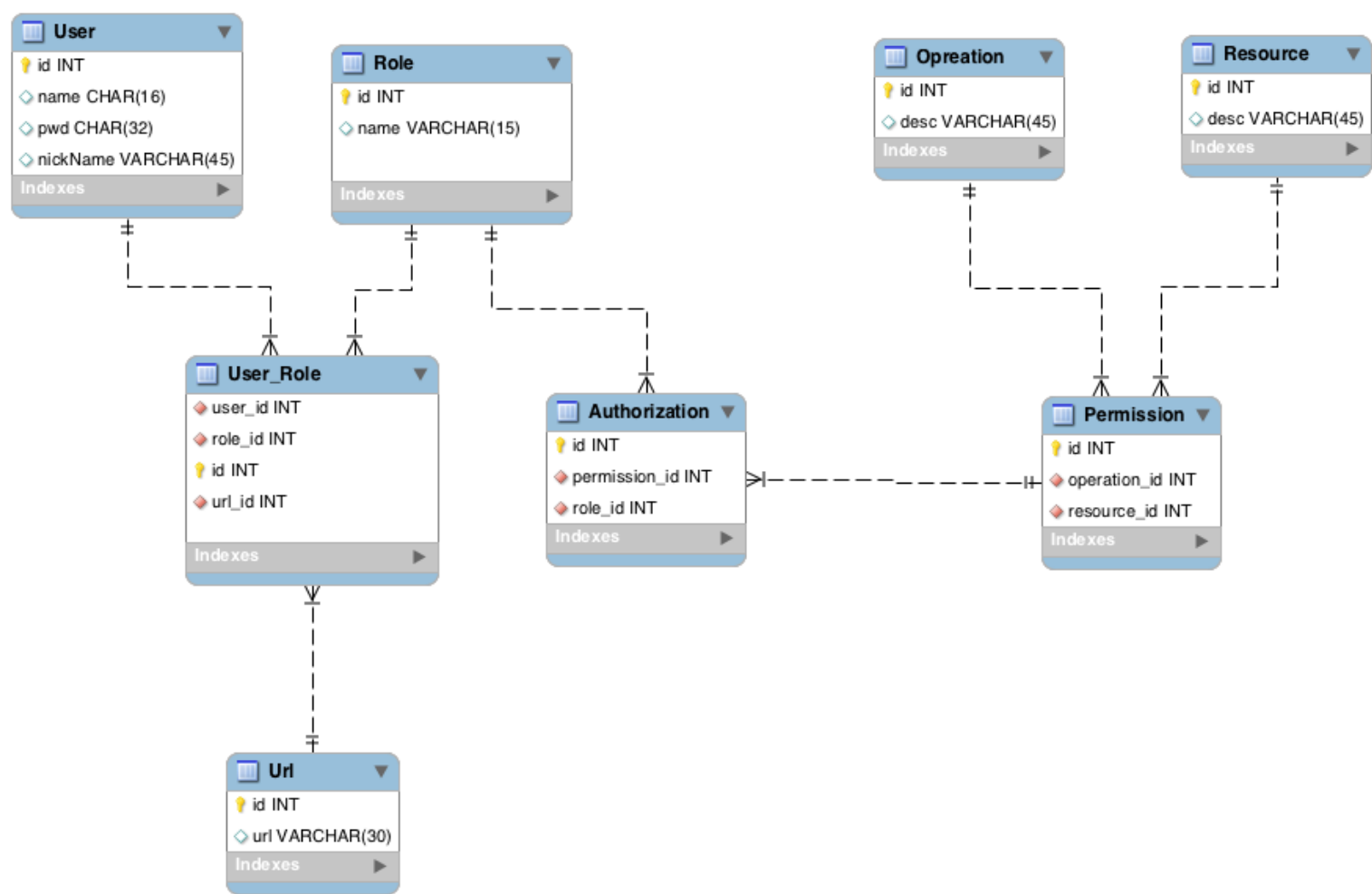
使用 RBAC 系统的好处在于将用户，角色，和权限分开，降低了系统的耦合度。由系统管理员进行角色的配置，将权限授予角色，将角色授予用户从而完成每个用户的权限管理。这样可以方便配置，也方便对单个角色，权限的管理。此时，RBAC 系统的模型图如下



下面，开始设计数据库。基于以上的场景分析，一个用户在不同场景下有不同的角色，但在一个场景下只有一个角色，而一个角色可以给不同用户。所以，用户和角色是一个多对多的关系，我们可以分

别设计一个用户表，角色表以及用户-角色表，其中用户-角色表存储的是用户在不同场景下与角色的对应关系。而一个角色可以有不同的权限，一个权限可以分给不同的角色，同样的，我们还需要设计一个角色表，一个权限表，和一个角色-权限表。而权限是由对象和操作组成的，所以我们还应该有操作表，对象表，以及他们构成的权限表。

画出所有的基本表结构如图所示



具体实现

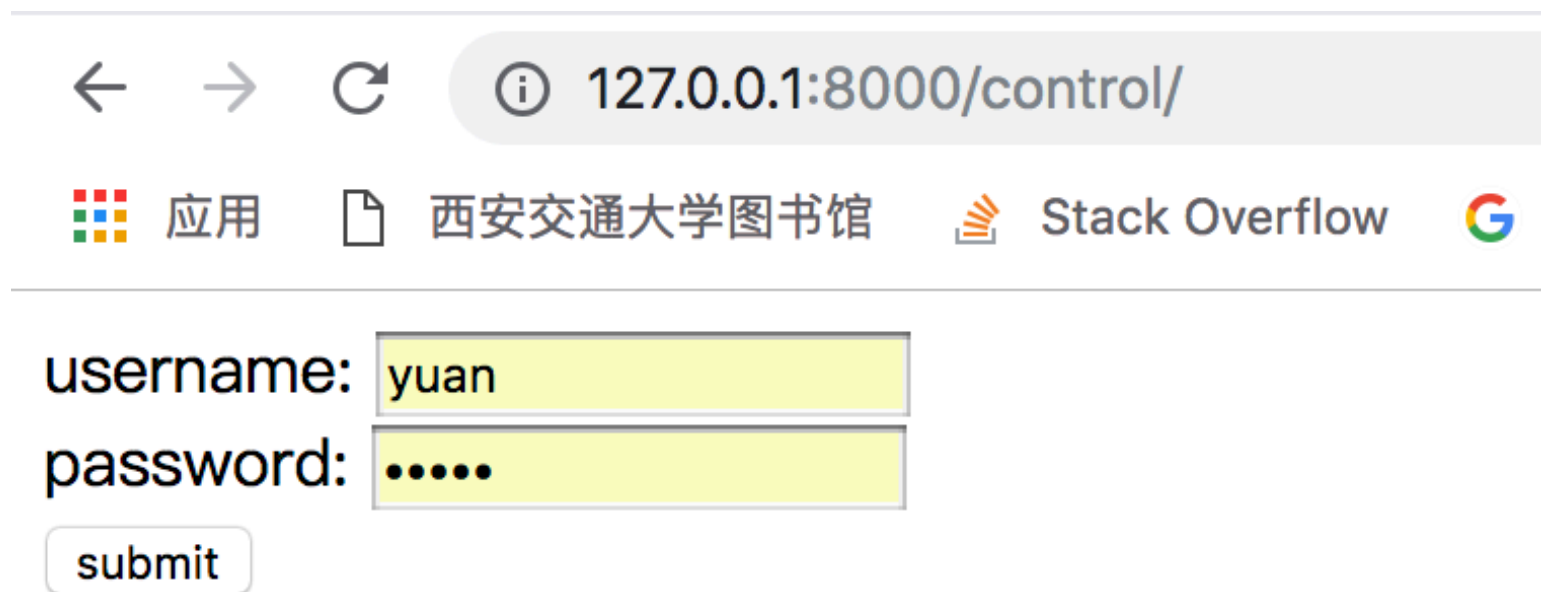
项目采用 django 框架，rbac 认证在后端完成，数据库采用了轻量级数据库 sqlite3，但只使用了 http 服务功能，登录，角色管理等部分都是手动实现。项目地址 (www.github.com/yuanjingsong/RBAC_demo) 系统主要实现了以下功能

1. 在用户未登录时，用户访问主页面会提示用户登录。(鉴权功能)
2. 在用户登陆后，根据用户访问的路由地址，给予用户不同的身份，同时不同的身份在同一个的页面有不同提示。

3. 对于不同身份的用户，系统会返回不同的权限（功能）

实现页面：

在未登录系统时，访问主页



← → ↻ ⓘ 127.0.0.1:8000/control/

应用 西安交通大学图书馆 Stack Overflow G

username: yuan

password:

submit

在登录之后，



← → ↻ ⓘ 127.0.0.1:8000/control/

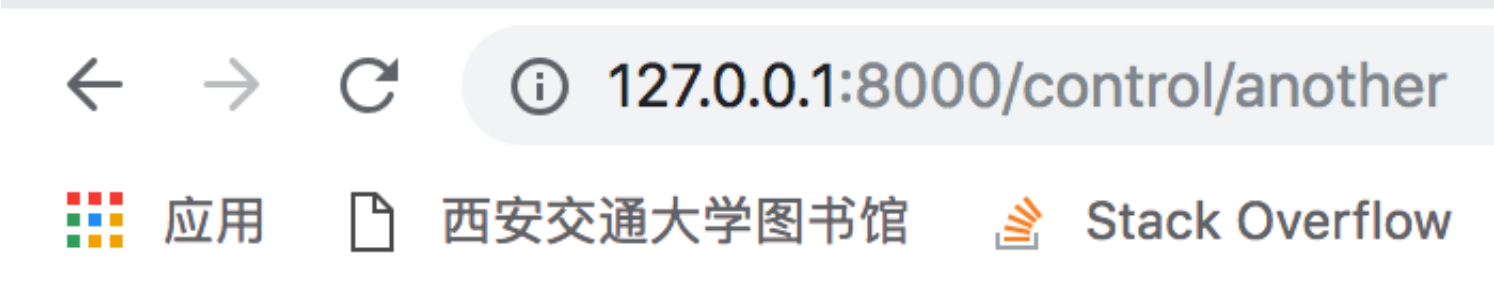
应用 西安交通大学图书馆 Stack Overflow G

Hello kk , Your role is boss

For management

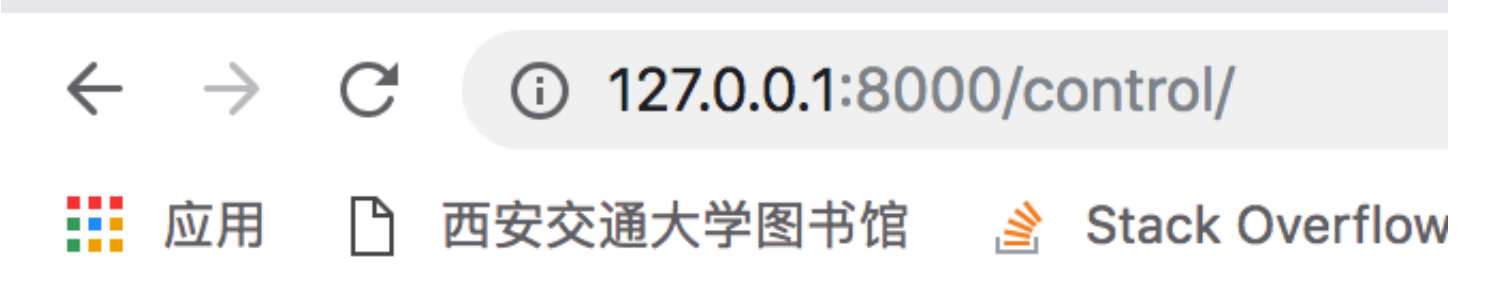
export report 退出系统

可以看到，我们以 boss 的身份，访问这个页面，同时此时具有导出成绩的功能。
但我们仍以原来的用户访问 `/control/another` 这个页面时，我们可以看到，其身份发生了变化，即实现了同一个用户在不同的页面角色控制。



Hello kk, Your Role is teacher

退出系统，当我们用另一个用户访问这个页面时



Hello Mr.Yuan , Your role is teacher

For correct homework

correct homework

退出系统



127.0.0.1:8000/control/



应用



西安交通大学图书馆



Stack Overflow

Hello Yuanjingsong , Your role is student

For doing homework

upload homework

退出系统

可以看到，当登录用户的用户身份是 student，只有提交作业的权限，而当登录的用户身份是 teacher 时，拥有了批改作业的权限，当登录的身份是 boss 的时候，拥有了导出成绩的功能。而针对不同角色的权限，都可以在数据库中配置，通过获取 Auth 表，进而渲染，动态在前端生成。

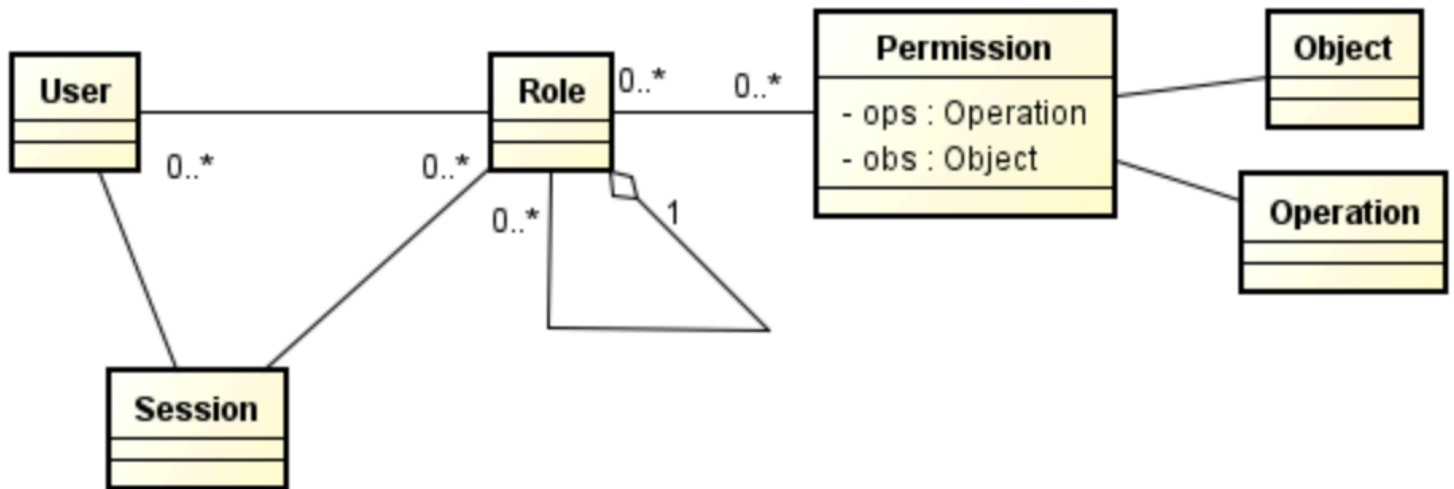
RBAC 的延伸模型

RBAC 不仅仅是只有这一种模型，在此思想的基础上，还延伸了其他的 RBAC 多级模型，分别是 RBAC1 级模型，RBAC2 级模型以及 RBAC3 级模型。

RBAC1 级模型

RBAC1 级模型在最基本的 RBAC 模型的基础上，加入了角色分级的概念，即将一些角色的权限抽象出来，对角色进行分级，形成了子类与父类关系。例如一个企业业务管理系统中，主任，科长，部长可能都同时具有相同一部分的权限，而此外，科长又具有比主任更高部分的权限，部长又具有比科长更高部分的权限。继承的关系为低级别的为父类，高级别的为子类。这种继承关系可以避免出现由于配置管理员的操作失误导致逻辑上应具有更高权限的人没有权限这种情况。同时，也更便于管理角色。

此时，系统的模型图如下



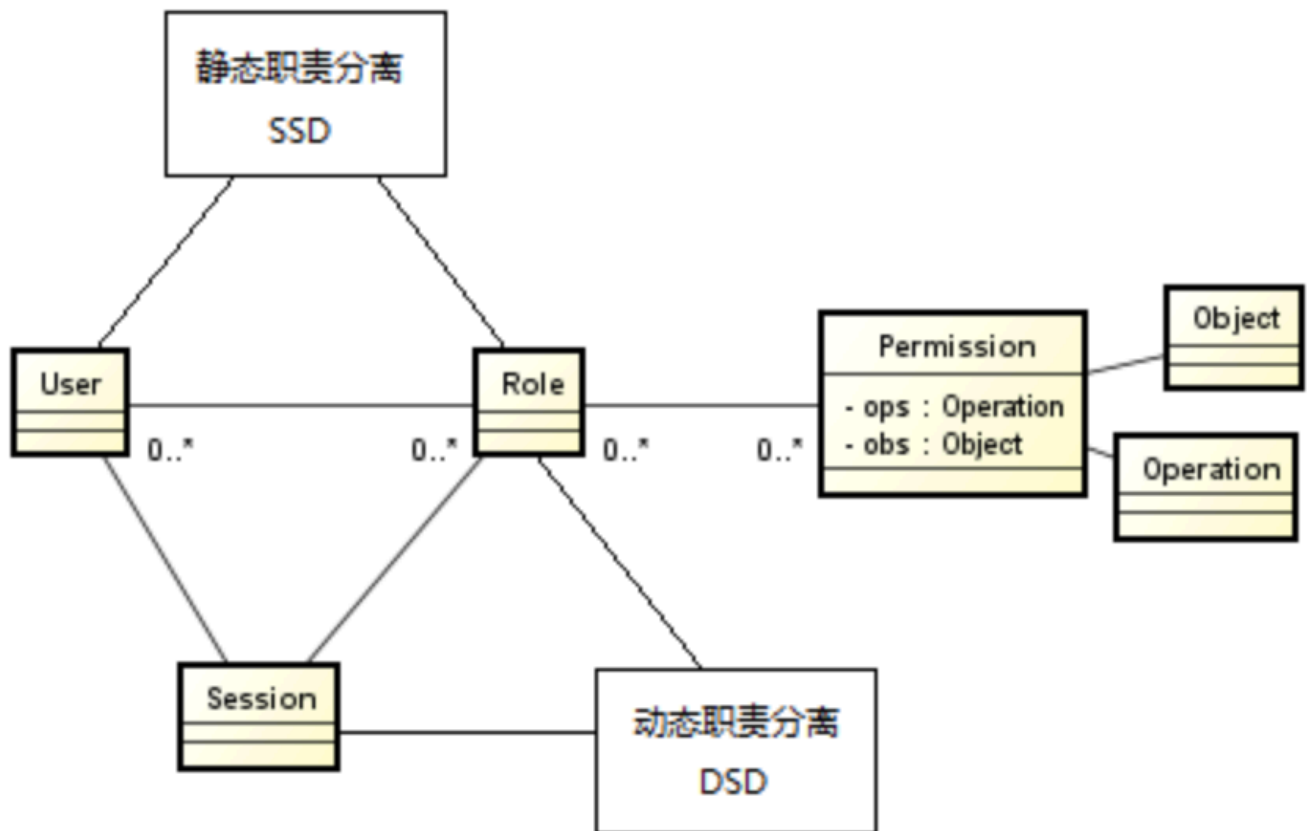
RBAC2 级模型

RBAC2 级模型在最基本的 RBAC 模型的基础上，加入了对角色的约束关系，例如角色的互斥关系，基数关系。实现了责任分离原则。这里的责任分离是确保安全性的几项基本原则。责任分离的原则简而言之即对敏感数据的操作由几个互斥角色共同完成。例如在企业中的财务管理，分为了出纳和会计两个角色，两个角色互斥，分管资金与账本，不允许出现两个角色同为一个人，保证了资金的安全。责任分离又可以分为静态责任分离与动态责任分离。

静态责任分离是指：在用户指定角色时就对角色施加约束。表现在为一个用户指定一个角色时，那么对这个用户而言就不能再任命为与该角色互斥的其他角色；某个角色的基数限制，比如某世界一流学府的校长只能为 1 人。

动态责任分离是指：在用户处于活跃状态使用系统时发挥的约束，与 session 相关。如一个用户可以在这个项目充当出纳角色，在另一个不相干的项目中充当会计角色，此时有两个不同的 session 场景，但不允许用户在同一个项目中既是出纳又是会计。

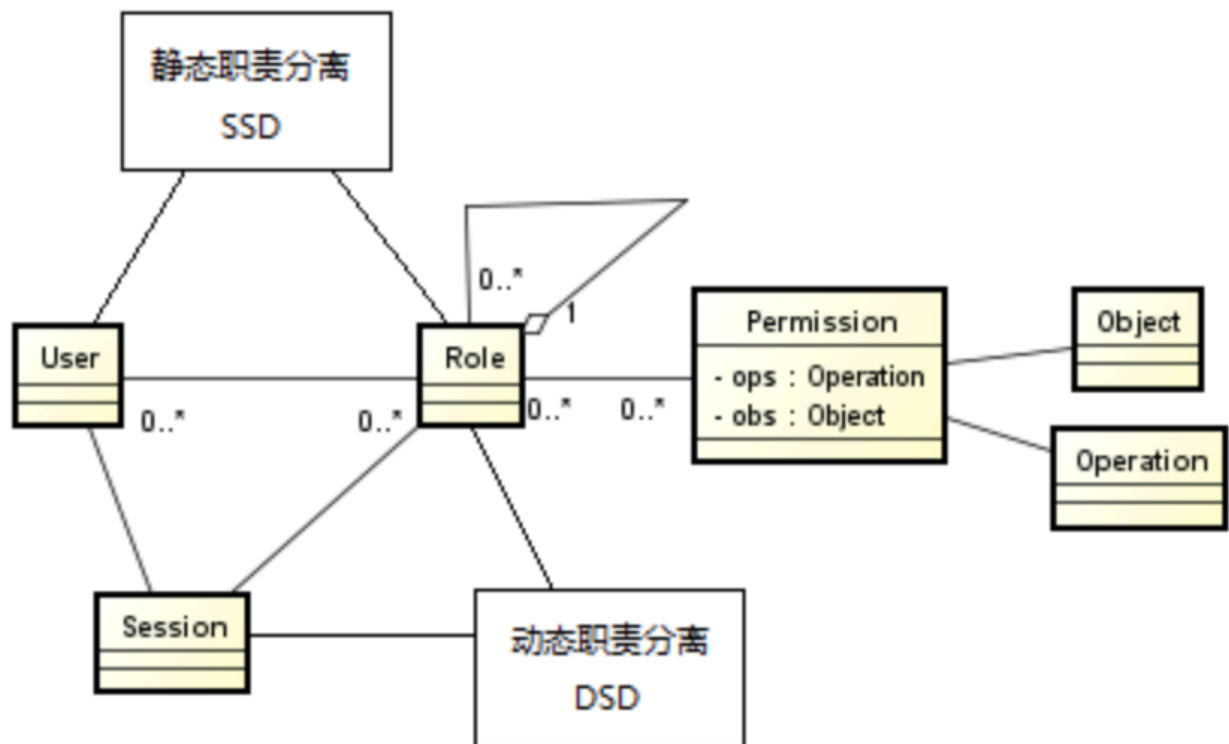
此时，系统的模型图如下



RBAC3 级模型

RBAC3 级模型同时引入了RBAC1 级模型和RBAC2 级模型。通过上述对 RBAC1 级模型和 RBAC2 级模型的描述，我们可以看到 RBAC1 级模型和 RBAC2 级模型是在两个不同的方向对 RBAC 模型进行扩展，RBAC1 级模型扩展的是角色的分类，引入了角色继承的概念。RBAC2 级模型扩展的是角色的约束。在 RBAC3 级模型中，是对 RBAC1 级模型和 RBAC2 级模型进行合并，由于 RBAC1 级模型和 RBAC2 级模型的拓展方向不是正交的，所以可能带来其一些问题，如对于角色的职责分离原则是否应该跟随角色的继承，如果一个角色有自身的静态约束，那么如果一个新角色是原角色的一个子类，是否应该继承这个静态约束。例如，一个职位可能有数量的约束，那么其子类是否也有数量的约束。

此时，系统的模型图如下：



RBAC 可能带来的一些问题和思考

多表查询的效率问题

在一个 rbac 系统中，实现的时候，我们可以看到，用户输入的只有用户名，密码和访问路由，而为了获得用户权限，我们需要通过用户名和访问路由查询一个角色控制表，显然这个表可能会非常大，它存的有不同用户在不同路由下的不同角色，在获取到角色后，我们需要继续查询这个角色所拥有的权限表，而权限表又是一个角色-权限表，如果需要细分到记录对不同资源拥有的权限，我们需要进一步查询资源-操作表，在这一系列的多表查询后，我们才可以返回用户所具有的具体操作。这在高并发时，对数据库查询作出了一个巨大的挑战。

在大规模用户下，可不可以放弃角色

rbac 系统主要是控制资源访问和数据访问。如果一个系统的普通用户非常多，那么对于大部分场景而言，都是同一个角色的操作，此时，可不可以放弃角色这个概念，仅针对少数的管理员赋予特殊权限，达到权限控制的效果，同时也可以对系统效率作出优化。

在 RBAC1 级模型下的继承问题

角色继承是否会继承了一些不必要的权限，例如高等级的角色继承了一些其应该拥有但基于业务场景

永远不会使用的权限，这个时候是否应该维护这些权限。

在继承关系中，如何设计数据库，才可以既体现子类与父类权限的差异性，又可以保证在子类在权限作出变动时，父类权限也可以做到相应的变更，如果仅仅是在 `role_permission` 表中对于不同的角色设置不同的权限，且对于有继承关系的角色对于同样的事务设置了同样的权限，那么这个继承仅仅存在在逻辑关系，并没有简化操作，此时这个继承是否有必要。

没有提供操作顺序的控制

RBAC 系统并没有对一个事务流的顺序作出一个严格的规定，比如在一个具有严格事务流的系统中，我们无法对其事务的先后顺序作出限制，所以 RBAC 系统不适合业务场景，仅仅适合控制场景，具体的业务流程，应该有一套系统。