CoolShark商城
数据库与数据表设计

3. 后台管理员管理相关数据表设计

目录

CONTENTS

1. 数据表概览

数据表概览

- 后台管理员相关的数据表有：
  - ams_permission：权限，用于存储项目中的所有权限标识
  - ams_role：角色，各后台管理员将被分配为某1种或多种角色，各角色将具有不同的权限
  - ams_role_permission：角色权限关联，用于分配各角色的权限
  - ams_admin：管理员信息
  - ams_admin_role：管理员角色关联，用于为各管理员分配角色，进而使得各管理员具有不同的权限
  - ams_login_log：管理员登录日志

2. 设计思路

RBAC模型（1/6）

- RBAC模型（Role-Based Access Control：基于角色的访问控制）模型是20世纪90年代研究出来的一种新模型，但其实在20世纪70年代的多用户计算时期，这种思想就已经被提出来，直到20世纪90年代中后期，RBAC才在研究团体中得到一些重视，并先后提出了许多类型的RBAC模型。其中以美国George Mason大学信息安全技术实验室（LIST）提出的RBAC96模型最具有代表，并得到了普遍的公认。

RBAC模型（2/6）

- RBAC认为权限授权的过程可以抽象地概括为：Who是否可以对What进行How的访问操作，并对这个逻辑表达式进行判断是否为True的求解过程，也即是将权限问题转换为What、How的问题，Who、What、How构成了访问权限三元组，具体的理论可以参考RBAC96的论文，这里我们就不做详细的展开介绍，大家有个印象即可。

RBAC模型（3/6）

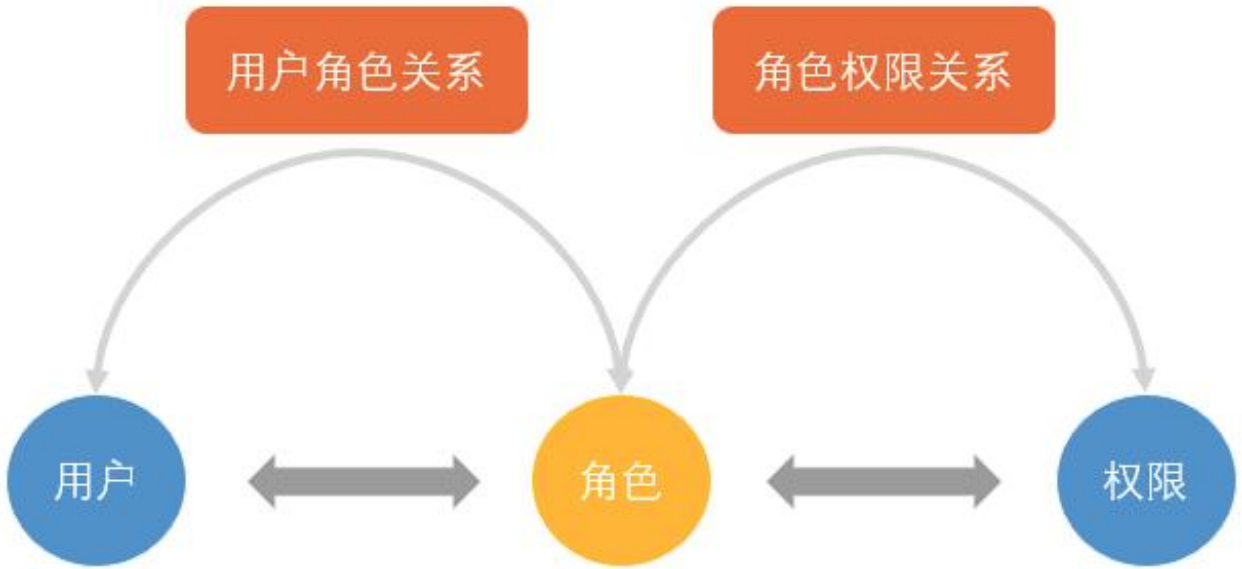
- 在RBAC模型里面，有3个基础组成部分，分别是：用户、角色和权限。
- RBAC通过定义角色的权限，并对用户授予某个角色从而来控制用户的权限，实现了用户和权限的逻辑分离（区别于ACL模型），极大地方便了权限的管理。

RBAC模型（4/6）

- RBAC模型中的核心概念包括：
  - User（用户）：每个用户都有唯一的UID识别，并被授予不同的角色
  - Role（角色）：不同角色具有不同的权限
  - Permission（权限）：访问权限
  - 用户-角色映射：用户和角色之间的映射关系
  - 角色-权限映射：角色和权限之间的映射

RBAC模型（5/6）

- RBAC模型中各概念的关系如下：



RBAC模型（6/6）

- RBAC支持三个著名的安全原则：最小权限原则、责任分离原则和数据抽象原则：
  - 最小权限原则：RBAC可以将角色配置成其完成任务所需的最小权限集合
  - 责任分离原则：可以通过调用相互独立互斥的角色来共同完成敏感的任务，例如要求一个计账员和财务管理员共同参与统一过账操作
  - 数据抽象原则：可以通过权限的抽象来体现，例如财务操作用借款、存款等抽象权限，而不是使用典型的读、写、执行权限

具体设计分析（1/5）

- 在主流的权限控制设计思路中，各种权限都会使用某个不重复的字符串表示，可以称之为权限标识，它们被存储中ams_permission表中，这些权限标识并不会直接赋予到任何一个管理员账号上，而是与“角色”（在某些软件中也称之为“用户组”或类似名称）进行关联，就有了ams_role和ams_role_permission表，所以，真正能让每个管理员账号具有不同权限的做法是“为每个管理员分配不同的权限”，就有了ams_admin_role表。

具体设计分析（2/5）

- 在具体的数据处理过程中，你将需要理解这几张表的核心价值，并掌握关联表查询，才可以查询到任何一个管理员账号所具备的权限，示例SQL语句如下（为了便于理解以上代码中的各个部分，以下SQL语句中各表、字段均未定义别名）：

```
select distinct ams_permission.value from ams_permission
left join ams_role_permission on ams_role_permission.permission_id=ams_permission.id
left join ams_role on ams_role_permission.role_id=ams_role.id
left join ams_admin_role on ams_admin_role.role_id=ams_role.id
left join ams_admin on ams_admin_role.admin_id=ams_admin.id
where ams_admin.id=1
order by ams_permission.value;
```

具体设计分析（3/5）

- 至于ams_login_log表，是用于记录管理登录日志的，并不是核心的或必要的，在当前项目中，设计这张表的主要目的是为了给同学们提供一些设计思想，在实际项目中，类似的日志表是存在的，越大、越规范的项目，日志表越多。

具体设计分析（4/5）

- 当你仔细查看数据表结构时，你会发现在ams_admin也存在与登录日志相关的字段，例如last_login_ip（最后登录IP地址）、login_count（登录次数）、gmt_last_login（最后登录时间），很显然，当有了ams_login_log表时，在ams_admin中的这3个字段并不是必要的，甚至是多余的，因为只需要对ams_admin和ams_login_log这2张表进行关联查询，就可以查询到任何管理员的最后登录信息，但是，有了3个字段后，只需要查询ams_admin这1张表，就可以做到了，虽然占用了更多的存储空间把同样的数据存储了2次，但是可以提升查询效率，这是典型的“牺牲空间，换取时间”的做法。

具体设计分析（5/5）

- 需要注意一点，根据阿里巴巴的设计规范：表达是与否概念的字段，必须使用is_xxx的方式命名，数据类型是unsigned tinyint (1表示是，0表示否)。在ams_admin表中，就有is_enable字段表示“是否启用”，由于值只可能是0或1，所以数据类型设计成tinyint。其实，无符号的unsigned tinyint最大值是255，由于实际值只可能是0或1，在某些设计里也会是unsigned tinyint(1)以将其取值范围设计得更小一些，一般来说，这样设计的问题不大，但是，目前有些框架会自动将tinyint(1)映射为Boolean类型，可能产生一些不必要的麻烦，而tinyint则映射为Integer，与设计本意是一致的！所以，在本项目中，使用tinyint时并不会添加(1)来约束取值。

谢谢！