数据库应用课程项目

ztxbbs

实验报告

张天翔

16307130026

目 录

1 课程项目意义与选题动机

1.1 项目意义

数据库引论一课详细讲解了数据库中各类概念，让我们对数据库的实现原理有了基本认识。同时，这门课也介绍关系数据库语言SQL。然而学会这些离动手做出数据库大项目来说还有差距。课程项目可以让我们加深对数据库的理解，增加我们对数据库的熟练度。

1.2 选题动机

本次课程项目，我选择做一个轻量的论坛。一个较为完整的论坛包括用户系统、管理系统、查询系统、用户间的关系、用户与帖子的关系等，这些关系较为复杂，可以很强地锻炼数据库的应用能力和代码能力。

2 项目整体规划

2.1 论坛结构选择

作为一个论坛，为了方便各种设备的访问，采用网页方式提供内容，因此以web应用的形式实现。

若之后有机会对项目进一步完善，会编写完整api，以对客户端提供数据交换。

2.2 技术栈

之前编写过类似的web应用，用过python+flask与nodejs+express。经过思考，选择使用nodejs来实现web应用。nodejs让javascript成为服务器脚本语言，实现前后端的统一，便于个人独立完成整个项目。近年来nodejs越来越热，生态圈愈发完善，本次项目使用nodejs也可进一步实现对此语言的熟悉。

html页面方面，使用了ejs模板。

对于数据库，使用较为常用且比较适合的MySql。同时，高级语言对数据库的查询优于数据库的原生查询，因此引入sequelize对nodejs的mysql进一步封装，以简化代码。

对于帖子内容的搜索，可以使用elasticsearch进行真正的模糊搜索，但是elasticsearch毕竟不是真正的数据库，并且为了减少编程的复杂程度，不打算实现真正的模糊查找，而使用正则表达式来搜索。

并没有使用前后端分离的方法。虽然nodejs有着一个功能强大的前端框架vue，但是作为独立编程的项目，前后端整合较为复杂，且不利于短时间内完成项目。因此放弃了后端express提供api加前端vue的架构。对于前端代码，jquery定然是必不可少的，同时引入了axios来实现ajax功能，实现后台与服务器api间的数据传递与前端页面的更新。

整体技术栈：nodejs+express+ejs+mysql+sequelize+axios

2.3 功能规划

用户系统

用户(user)
注册(signin)
登录(signup)
登出(signout)
设置(edit)
邮箱认证(confirm)

私信(message)
关注(follow)
关注者(follower)
黑名单(blacklist)

主题系统

论坛(community)
板块(forum)
主题(topic)
回复(comment)
的添加、删除、修改、查找

管理系统

系统通知(notices)
系统配置(settings)

3 数据库设计

3.1 ER图

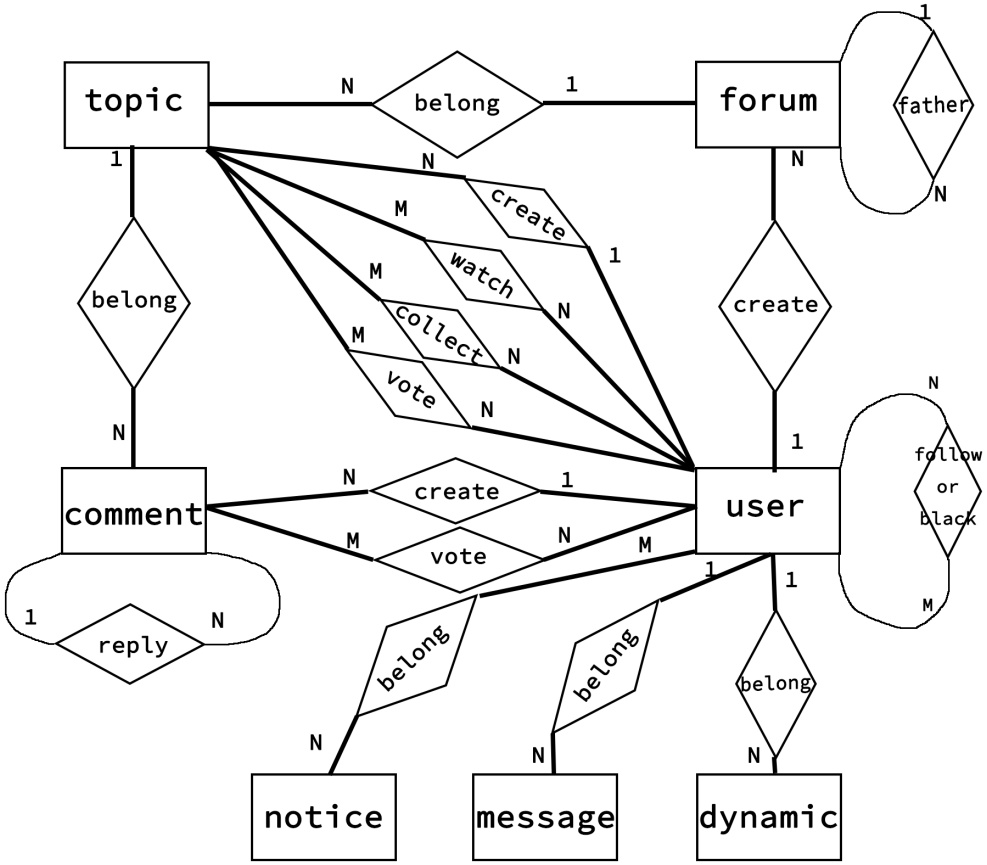

图2

3.2 关系模式

User(id, username, nickname, password, email, sex, birthday, website, country, city, description, raw_introduction, introduction, raw_signature, signature, avatar, cover, authority_user, authority_admin, register_time, last_login_time, count_topics, count_followers, count_followings, confirm_code, confirm_date, count_messages)

Forum(id, user_id, forum_id, title, description, authority_topic, cover, sticky, last_reply_topic, last_reply_tid, last_reply_time, last_reply_nickname, last_reply_uid, count_topics)

Topic(id, user_id, forum_id, title, raw_content, content, authority_comment,

cover, sticky, create_time, nickname, last_reply_time, last_reply_nickname, last_reply_uid, last_edit_time, count_, count_reply, count_watch, count_collect, count_vote)

Comment(id, user_id, topic_id, raw_content, content, abstract, create_time, nickname, reply_uid, reply_nickname, reply_abstract, last_edit_time, count_edit, count_vote)

Message(id, user_id, raw_content, content, abstract, create_time, type, other_id, other_nickname, other_avatar, read)

Dynamic(id, user_id, header, content, create_time, link, show_avatar, show_id, read)

Notification(id, user_id, raw_content, content, abstract, create_time, broadcast_time)

FollowOrBlack(userA_id, userB_id, type, create_time)

NotificationNotifyUser(notification_id, user_id, create_time, read)

UserCollectTopic(user_id, topic_id, create_time)

UserWatchTopic(user_id, topic_id, create_time)

UserVoteTopic(user_id, topic_id, create_time, num)

UserVoteComment(user_id, comment_id, create_time, num)

其中一些设计需要解释一下。

表中存在一些count属性，这些属性可以通过SQL聚合函数count得到，但是，这些属性常用于展示或用于分页，那么这些值就会被经常使用，因此，考虑到时间效率与用户体验，将其存为属性。这是以空间换时间。

表中存在一些last_reply属性，这些属性用于展示，更新较少而查询较多，因此以空间换时间将其存为属性。

表中存了一些uid与nickname，甚至有avatar，这也是以空间换时间。其中的uid 没有存为外键，是由于这个属性存在的意义不在于连接另一个user，而在于存储user的信息，因而它不需要满足参照完整性约束，也就不存为外键。

表中许多设计都是为了避免多表连接操作。当然N:M关系的链接操作定然是必不可少的。

4 路由设计

”*”代表着响应将是json格式的，这类路由与api功能相同，用于页面后台的数据交换。

”-”代表着路由存在，但功能尚待实现。

method	route	action
get	/	/community

4.1 用户

路由前缀为/users

method	route	action
get	/signin	登录页面
post	/signin	登录动作
get	/signup	注册页面
post	/signup	注册动作
get	/signout	登出动作
get	/confirm?confirm_code=	认证邮箱动作
post	/confirm	*发送认证邮件
get	/message	获取私信列表
get	/message/send?user_id=	发送私信
post	/message/send	发送私信动作
get	/message/:mid	私信详情页
post	/message/:mid/read	-*将私信设为已读
post	/message/:mid/unread	-*将私信设为未读
post	/message/:mid/remove	-*删除私信
post	/message/:mid/reply	-*回复私信
get	/friends	/users/friends/following
get	/friends/following	正在关注用户页
get	/friends/follower	关注者页
get	/friends/blacklist	黑名单页
get	/notice	-获取通知列表
post	/notice/:nid/read	-*将通知设为已读
post	/notice/:nid/unread	-*将通知设为未读
post	/notice/:nid/remove	-*删除通知
post	/follow?user_id=	*关注
post	/unfollow?user_id=	*取消关注
post	/ban?user_id=	*拉黑
post	/unban?user_id=	*取消拉黑
get	/:uid	用户uid的资料页
get	/:uid/edit	用户uid的编辑页
post	/:uid/edit	用户uid的编辑动作

4.2 论坛

路由前缀/community

method	route	action
get	/	/community/forums

4.3 动态

路由前缀/community/dynamic

method	route	action
get	/	获取动态页

4.4 板块

路由前缀/community/forums

method	route	action
get	/	/community/forums/1
get	/create_forum	创建版块页面
post	/create_forum	创建版块动作
get	/:fid	查看板块详情页面
get	/:fid/edit	板块编辑页
post	/:fid/edit	板块编辑动作
post	/:fid/remove	*板块删除动作

4.5 主题

路由前缀/community/forums/topics

method	route	action
get	/create_topic	添加主题页面
post	/create_topic	添加主题动作
get	/:tid	查看主题详情页面
get	/:tid/edit	主题编辑页
post	/:tid/edit	主题编辑动作
post	/:tid/remove	*主题删除动作
post	/:tid/watch	-*主题关注动作
post	/:tid/unwatch	-*主题取消关注动作
post	/:tid/collect	-*主题收藏动作
post	/:tid/uncollect	-*主题取消收藏动作
post	/:tid/voteup	-*主题评分+1动作
post	/:tid/votedown	-*主题评分-1动作

4.6 回复

路由前缀/community/forums/topics/comments

method	route	action
post	/create_comment	*添加回复动作
post	/:cid/edit	*回复编辑动作
post	/:cid/remove	*回复删除动作
post	/:cid/reply	-*回复回复动作
post	/:cid/voteup	-*回复评分+1动作
post	/:cid/votedown	-*回复评分-1动作

4.7 搜索

路由前缀/search

method	route	action
get	/	搜索页面
post	/?mode=	搜索动作

4.8 管理

路由前缀/manage

method	route	action
get	/	/manage/status
get	/status	-论坛状态页
get	/settings	-论坛设置页
get	/notices	-通知列表页
get	/notices/create	-通知新建页
post	/notices/create	-通知新建动作
get	/notices/:nid	-通知详情页
post	/notices/:nid/broadcast	-*通知广播动作

5 技术细节与难点

5.1 用户

用户登录支持邮箱与账号登录，因为注册时用户名不允许使用特殊字符，因此可以通过‘@’符号方便地判断登录方式。

用户注册时，先进行简单地账号密码邮箱的填写，通过检查后对邮箱发送邮件进行认证，同时重定向至个人资料编辑页进行进一步的账号信息完善。

其中，邮箱认证方法为，生成时间戳加随机字符串得到认证码，将其做成链接发送至需要认证的邮箱，邮箱通过此链接完成认证。如果用户更改邮箱，则会将用户重新设置为未认证用户。

用户拥有两种权限，一种是作为普通用户的权限，一种是作为管理用户的权限。将所有的权限细分，具体为，用户权限：是否被禁、认证状态，管理权限：所有权限、管理权限的权限、管理用户、管理系统配置、管理系统通知、管理用户的普通权限、管理板块、管理主题与回复。将这些细分后的权限设置以二进制的形式存储为两个整数，存到用户的属性中。这样做节省了一些空间并且利于权限的扩展。

对于用户，应用在每次启动时会检查是否存在root账号，并将其权限设置为所有权限，root账号的密码由配置设定。

5.2 板块

板块是对主题的分类，板块之间有从属关系。在显示板块时，不仅仅要显示当前板块下的主题，也要显示当前板块的子版块及其状态，这里用了递归的方法来实现。这样实现的原因是，板块子版块一般不会过多，因此可以递归得到其子版块展示出来。应用中实现了向下展示三级。

板块有着自己的信息，最后回复主题，最后回复时间等，这些信息由板块范围内主题及回复来更新。

板块有其权限，是否可以添加主题，此权限由拥有管理板块权限的用户设置。

所有板块中只有一个板块不存在父板块，它就是板块1，可称作根板块。它也在应用启动时被检查或创建。

板块下的帖子默认按照最后回复时间降序排序，通过选择，可以选择按照发布时间降序、评论数降序、阅读数降序。

5.3 主题

主题的内容采用标记文本markdown的语法来编写，其中也加入了代码块语法高亮，mathjax公式渲染。

主题也有其权限，可以被任意回复，或是被管理员回复，或是禁止回复。主题也有指定功能。这两项由主题管理员设置。

主题也有众多的信息。当主题被创建时，会更新所在板块的最后回复信息，并不断更新父板块的信息，直至不能更新或到达根板块。

5.4 回复

回复的内容语法与主题同。

回复被创建时会更新主题的最后回复信息等，进而更新板块的信息。

5.5 私信

用户对另一个用户发送私信，这个私信将被分成两份存储，一份归属于发送者，一份归属于接受者，同时用一个字段来判断是发出还是接收，于是两份私信的删除，互不干扰。如果发信人被收信人拉黑，则发送不成功。

5.6 动态

许多内容会产生动态，用一个字段来记录动态的类型过于繁琐，且不利于扩展，因此将动态格式化。

一个动态，必定是由人产生，因此每个动态展示一个头像；一个动态必定有一个标题；一个动态可能会有附加的解释；一个动态必有可以指向的链接。

产生动态的地方有一些，如：你关注的用户关注了一个用户、你关注的用户发布了新主题、有人回复了你、有人关注了你等等。

5.7 文件的存储

文件的存储使用了存路径的方式，本应用中的图片都是可公开的图片，因此将其存储在public目录中。

5.8 分页

许多列表用到了分页，分页需要统计结果的个数，然而作为web应用，通过聚合函数count得到记录数显然是不明智的，因此很多地方采用了存储count值到相关记录并不断更新其值。

例如板块下的主题、搜索的结果、消息列表等。

有一个难点就是，分页时的a标签都是用get方法访问url的，但是这样的方法会将查询的数据丢掉。解决办法就是将a标签的链接改为js函数，用js函数将form的action改掉page，然后让form提交即可。