济票坊——概要设计规约

一、引言

1.1 概要设计依据

- 杜庆峰老师上课所讲的概要设计基本原则
- 济票坊-需求分析规约

1.2 参考资料

《Spring Boot与微服务实战》

《Mastering Spring Boot 2.0》

《Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives》

《Domain-Driven Design: Tackling Complexity in the Heart of Software》

《Design Patterns: Elements of Reusable Object-Oriented Software》

1.3 假定和约束

本项目开发主要受软件工程课程的约束，因此本项目的假定和约束如下所示：

1. 项目开发期限为2个月，时间为2023年11月~12月。
2. 项目开发无经费，设备条件为4台Windows操作系统电脑以及阿里云云平台等。
3. 项目在开发前线下采访了4名社团管理人员，线上通过问卷调研的方式收集了49份问卷，并初步明晰了项目需求
4. 在交流过程中，我们每周两次线下汇报工作进度，同时通过Github进行代码协作管理。

二、概要设计

2.1 系统总体架构设计

2.2 系统软件结构设计

2.3 接口设计

2.3.1 注册登录子系统

registerStudent

- **方法签名：** `public ResponseEntity<?> registerStudent(@RequestBody RegStudentRequest registrationRequest)`
- **实际调用方法：** `registerService.registerStudent(registrationRequest)`
- **描述：** 用于注册学生用户。

registerSociety

- **方法签名：** `public ResponseEntity<?> registerSociety(@RequestBody RegSocietyRequest registrationRequest)`
- **实际调用方法：** `registerService.registerSociety(registrationRequest)`
- **描述：** 用于注册社团。

getUnauthSocieties

- **方法签名：** `public ResponseEntity<Page<Society>> getUnauthSocieties(@RequestParam(defaultValue = "0") int page, @RequestParam(defaultValue = "10") int pageSize)`
- **实际调用方法：** `societyService.getUnauthenticatedSocieties(page, pageSize)`
- **描述：** 获取未经验证的社团列表。

login

- **方法签名：** `public ResponseEntity<?> login(@RequestBody LoginRequest loginRequest)`
- **实际调用方法：** `loginService.login(loginRequest.getUsername(), loginRequest.getPassword())`
- **描述：** 用户登录。

getStudentProfile

- **方法签名：** `public ResponseEntity<?> getStudentProfile(@RequestParam String username)`
- **实际调用方法：** `profileService.getStudentProfile(username)`
- **描述：** 获取学生个人资料。

modifyStudentProfile

- **方法签名：** `public ResponseEntity<?> modifyStudentProfile(@RequestBody ModifyStuProfileReq modifyRequest)`
- **实际调用方法：** `profileService.modifyStudentProfile(modifyRequest)`
- **描述：** 修改学生个人资料。

getSocietyProfileInfo

- **方法签名：** `public ResponseEntity<?> getSocietyProfileInfo(@RequestParam String username)`
- **实际调用方法：** `profileService.getSocietyProfileInfo(username)`
- **描述：** 获取社团个人资料信息。

getSocietyProfileLogo

- **方法签名：** `public ResponseEntity<?> getSocietyProfileLogo(@RequestParam String username)`
- **实际调用方法：** `registerService.getStudentProfile(username)`
- **描述：** 获取社团的Logo。

getSocietyProfileImages

- **方法签名：** `public ResponseEntity<?> getSocietyProfileImages(@RequestParam String username)`
- **实际调用方法：** `registerService.getStudentProfile(username)`
- **描述：** 获取社团的图片。

modifySocietyProfile

- **方法签名：** `public ResponseEntity<?> modifySocietyProfile(@RequestParam String username, @RequestBody ModifyStuProfileReq modifyRequest)`
- **实际调用方法：** `registerService.modifyStudentProfile(username, modifyRequest)`
- **描述：** 修改社团个人资料。

2.3.2 活动管理与报名子系统

getActivityPage

- 方法签名：public ResponseEntity<?> getActivityPage(@PathVariable("page") int page)
- 实际调用方法：activityPersonalService.getActivityPage(page)

getActivity

- 方法签名：public ResponseEntity<?> getActivity(@PathVariable("actId") int actId)
- 实际调用方法：activityPersonalService.getActivity(actId)

addBrowse

- 方法签名：public ResponseEntity<?> addBrowse(@RequestBody AddBrowseRequest addBrowseRequest)
- 实际调用方法：activityPersonalService.addBrowse(addBrowseRequest)

getFavour

- 方法签名：public ResponseEntity<?> getFavour(@PathVariable("stuId") int stuId)
- 实际调用方法：activityPersonalService.getFavour(stuId)

addFavour

- 方法签名：public ResponseEntity<?> addFavour(@RequestBody FavourRequest favourRequest)
- 实际调用方法：activityPersonalService.addFavour(favourRequest)

deleteFavour

- 方法签名：public ResponseEntity<?> deleteFavour(@RequestBody FavourRequest favourRequest)
- 实际调用方法：activityPersonalService.deleteFavour(favourRequest)

getIndentPage

- 方法签名：public ResponseEntity<?> getIndentPage(@RequestBody GetIndentPageRequest getIndentPageRequest)
- 实际调用方法：activityPersonalService.getIndentPage(getIndentPageRequest)

getIndent

- 方法签名：public ResponseEntity<?> getIndent(@PathVariable("indId") int indId)
- 实际调用方法：activityPersonalService.getIndent(indId)

addIndent

- 方法签名：public ResponseEntity<?> addIndent(@RequestBody AddIndentRequest addIndentRequest)

- 实际调用方法：activityPersonalService.addIndent(addIndentRequest)

cancelIndent

- 方法签名：public ResponseEntity<?> cancelIndent(@RequestBody CancelIndentRequest cancelIndentRequest)
- 实际调用方法：activityPersonalService.cancelIndent(cancelIndentRequest)

writeOffIndent

- 方法签名：public ResponseEntity<?> writeOffIndent(@PathVariable("indId") int indId)
- 实际调用方法：activityPersonalService.writeOffIndent(indId)

changeIndentNotes

- 方法签名：public ResponseEntity<?> changeIndentNotes(@RequestBody ChangeIndentNotesRequest changeIndentNotesRequest)
- 实际调用方法：activityPersonalService.changeIndentNotes(changeIndentNotesRequest)

2.3.3 活动咨询子系统

getCommentByActId

- 方法签名：public ResponseEntity<?> getCommentByActId(@PathVariable("actId") int actId)
- 实际调用方法：communicateService.getCommentByActId(actId)

addComment

- 方法签名：public ResponseEntity<?> addComment(@RequestBody AddCommentRequest addCommentRequest)
- 实际调用方法：communicateService.addComment(addCommentRequest)

replyComment

- 方法签名：public ResponseEntity<?> replyComment(@RequestBody ReplyCommentRequest replyCommentRequest)
- 实际调用方法：communicateService.replyComment(replyCommentRequest)

rateActivity

- 方法签名：public ResponseEntity<?> rateActivity(@RequestBody RateActivityRequest rateActivityRequest)
- 实际调用方法：communicateService.rateActivity(rateActivityRequest)

changeRating

- 方法签名：public ResponseEntity<?> changeRating(@RequestBody RateActivityRequest rateActivityRequest)
- 实际调用方法：communicateService.changeRating(rateActivityRequest)

2.3.4 秩序管理子系统

getAppealPage

- 方法签名：public ResponseEntity<?> getAppealPage(@PathVariable("page") int page)
- 实际调用方法：orderService.getAppealPage(page)

getAppeal

- 方法签名：public ResponseEntity<?> getAppeal(@PathVariable("appId") int appId)
- 实际调用方法：orderService.getAppeal(appId)

addAppeal

- 方法签名：public ResponseEntity<?> addAppeal(@RequestBody AddAppealRequest addAppealRequest)
- 实际调用方法：orderService.addAppeal(addAppealRequest)

setUserProhibitedStatus

- 方法签名：public ResponseEntity<?> setUserProhibitedStatus(@RequestBody SetUserProhibitedStatusRequest setUserProhibitedStatusRequest)
- 实际调用方法：orderService.setUserProhibitedStatus(setUserProhibitedStatusRequest)

2.4 界面设计

2.5 数据库设计

2.5.1 数据库的逻辑结构设计

在数据模型设计阶段完成后，结合需求分析阶段对数据需求的深入分析，我们对整个系统的数据库逻辑结构进行了精细化设计。这包括系统实体及其关联的综合分析，并据此构建了系统的综合E-R图（实体-关系图）。详细描述如下：

2.5.1.1 数据库实体设计

在构建数据库的物理存储模型前，我们首先对数据库的各个实体进行了定义和规划。具体实体列表如下所示：

1. <entity>用户：

是指使用本社团活动管理系统的人，通常拥有一个用户账号，用户统一使用用户名登录。在本系统中，用户泛化为个人用户和社团管理员和系统管理员。根据用户类型的不同，用户有包括但不限于报名活动，管理活动等操作。系统会记录用户的登录状态，以便于提供个性化服务。用户的校区信息用于活动推荐系统的就近推荐，余额用于支付活动。用户的注册时间可以帮助我们了解用户在平台上的活跃度和历史行为。

实体名称	用户
标识符	user
描述	记录用户的账号和各种账号基本信息
主键	用户ID
数据项（属性）	用户ID，用户名，密码，邮箱，电话号码，所在校区，登录状态，账号状态，账户余额，支付体现密码，注册时间，用户类型

2. <entity>社团活动关键词：

是指在本社团活动管理系统中，系统、个人用户和社团管理员共同维护的用于进行活动特点描述的关键词。

实体名称	用户
标识符	keywords
描述	用于进行活动特点描述的关键词
主键	活动类型关键词
数据项（属性）	活动类型关键词

3. <entity>个人用户：

是指在本社团活动管理系统中浏览和报名活动的用户。每个个人用户都有一个用户账号，通常以用户名进行识别。他们可以进行浏览活动，报名活动，发表评论等操作。系统会记录个人用户的登录状态，以便于提供个性化服务。个人用户可以在本系统内充值，用户余额用于支付活动报名费。

实体名称	个人用户
标识符	student
描述	记录个人用户的账号和各种学生基本信息
主键	个人用户ID
数据项（属性）	个人用户ID，学生姓名，学生所在年级，学生所在学院，学生所在专业，个人备注，个人活动喜好关键词

4. <entity>社团管理员：

是指在本社团活动管理系统中发布和管理活动的用户。每个社团都有一个社团管理员账号，通常以用户名进行识别。他们可以进行发布活动，管理活动，处理报名表单等操作。系统会记录社团管理员的登录状态，以便于提供个性化服务。社团管理员的校区位置、活动类别等信息综合之后，系统会整合这些信息来为个人用户推荐活动。

实体名称	社团管理员用户
标识符	society
描述	记录社团管理员的账号和各种社团基本信息
主键	社团管理员用户ID
数据项（属性）	社团管理员用户ID，社团名称，社团介绍，社团类别，社团Logo，社团活动关键词，社团图片

5. <entity>社团负责人：

是指在本社团活动管理系统中可联系的社团负责人。每个社团都有一个或多个社团负责人，通常以社团管理员用户ID和社团负责人学号进行识别。

实体名称	社团负责人
标识符	society_admin
描述	记录社团负责人的基本信息，可由社团管理员用户维护
主键	社团管理员用户ID，社团负责人学号
数据项（属性）	社团管理员用户ID，社团负责人学号，社团负责人姓名，社团负责人电话号码，社团负责人邮箱

6. <entity>系统管理员（待设计）：

是指在本社团活动管理系统中进行管理和维护操作的用户。管理员有权进行高级操作，包括但不限于处理用户投诉，管理个人用户和社团管理员，编辑系统公告等。每个管理员都有一个用户账号，通常以管理员ID进行识别。系统会记录管理员的登录状态。

实体名称	系统管理员
标识符	admin
描述	记录系统管理员的账号和基本信息
主键	系统管理员用户ID

数据项（属性）	系统管理员用户ID，xxx

7. <entity>社团活动：

是指在本社团活动管理系统中由社团管理员发布并供个人用户报名的活动。每个活动都有一个唯一的活动ID，由系统自动生成。

实体名称	社团活动
标识符	activity
描述	记录活动的基本信息，可由社团管理员用户维护
主键	活动ID
数据项（属性）	活动ID，活动名称，活动简介，活动位置，活动报名费，活动发布时间，活动开始报名时间，活动截止报名时间，活动时间，活动总容量，活动剩余名额，活动评分，评分人数，活动发布社团id，活动关键词，活动图片

8. <entity>活动报名订单：

是指在本社团活动管理系统中个人用户报名活动后生成的记录，在本系统中，订单与活动之间是多对一的关系，每一个订单都只和一种活动相对应。每个订单都有一个唯一的订单ID，由系统自动生成。在本系统中，订单实体还泛化出“已取消订单”，它多出了“取消原因、取消时间、退款金额”三个属性，用于记录取消订单的信息。

实体名称	活动报名订单
标识符	indent
描述	记录个人用户报名一次活动的报名记录。用户完成一次报名后，生成一项新的订单实体
主键	订单ID
数据项（属性）	订单ID，支付金额，创建时间，活动参与核销码，报名者姓名，报名者学号，用户备注，订单状态，评分，活动ID，学生账号ID，社团账号ID，退款时间，退款原因，退款金额

9. <entity>评论：

是指在本社团活动管理系统中，个人用户对报名的活动进行评价和反馈的记录。每个评论都有一个唯一的评论ID，由系统自动生成。

实体名称	评论

标识符	comment
描述	记录用户发布的评论的基本信息。
主键	评论ID
数据项（属性）	评论ID，父评论，评论内容，评论时间，活动ID，用户ID

10. <entity>申诉：

是指在本社团活动管理系统中，用户（个人用户或社团管理员）对某些决定或行为（如恶意刷单行为、被投诉的评论、被撤销的活动发布资格等）提出反馈或异议的记录。每个申诉都有一个唯一的申诉ID，由系统自动生成。

实体名称	申诉
标识符	appeal
描述	记录用户发布的申诉的基本信息。
主键	申诉ID
数据项（属性）	申诉ID，申诉时间，申诉事项，申诉描述，申诉用户ID，申诉活动ID，申诉评论ID，发起申诉用户ID，申诉图片

11. <entity>公告：

是指在本社团活动管理系统中由社团管理员发布的信息，用于向个人用户传达重要信息或通知。每个公告都有一个唯一的公告ID，由系统自动生成，系统管理员可以改变它的显示状态。

实体名称	公告
标识符	appeal
描述	记录社团管理员发布的公告的基本信息。
主键	申诉ID
数据项（属性）	申诉ID，申诉时间，申诉事项，申诉描述，申诉用户ID，申诉活动ID，申诉评论ID，发起申诉用户ID

12. <entity>公告：

是指在本社团活动管理系统中由社团管理员发布的信息，用于向个人用户传达重要信息或通知。每个公告都有一个唯一的公告ID，由系统自动生成，系统管理员可以改变它的显示状态。

实体名称	公告
标识符	appeal
描述	记录社团管理员发布的公告的基本信息。
主键	申诉ID
数据项（属性）	申诉ID，申诉时间，申诉事项，申诉描述，申诉用户ID，申诉活动ID，申诉评论ID，发起申诉用户ID

13. <entity>收藏记录：

是指在本社团活动管理系统中由个人用户在收藏活动时产生的收藏记录。当个人用户在平台上收藏活动时，系统会在收藏记录表中生成相应的记录。

实体名称	收藏记录
标识符	favour
描述	记录个人用户产生的收藏的基本信息。
主键	活动ID，收藏者ID
数据项（属性）	活动ID，收藏者ID

14. <entity>浏览记录：

是指在本社团活动管理系统中由个人用户在浏览活动时产生的浏览记录。当个人用户在平台上浏览活动信息页面时，系统会在浏览记录表中生成相应的记录。

实体名称	浏览记录
标识符	browse
描述	记录个人用户产生的浏览记录的基本信息。
主键	活动ID，浏览者ID，起始浏览时间
数据项（属性）	活动ID，浏览者ID，起始浏览时间，报名情况

15. <entity>聊天记录：

是指在本社团活动管理系统中当个人用户向社团管理员进行活动咨询时产生的聊天记录。

实体名称	聊天记录
标识符	chat
描述	记录个人用户与社团管理员产生的聊天记录的基本信息。
主键	个人用户ID，社团管理员ID，聊天时间
数据项（属性）	个人用户ID，社团管理员ID，聊天时间，聊天内容，发出者

16. <entity>活动场地（待设计）：

是指在

17. <entity>礼券（待设计）：

是指

2.5.1.2 数据库关系设计

基于前述实体，我们深入分析了实体间的相互关系。鉴于存在强实体集和弱实体集，实体间的关系设计需特别细致，以便为之后的物理模型构建及数据库范式的遵循打下基础。以下列出了数据库中定义的各类关系集：

1. 个人用户和社团管理员的**聊天**关系：

当个人用户与社团管理员之间发生互动（如提问，询问活动详情等），系统会在聊天记录表中生成相应的记录，描述性属性包括聊天内容，时间戳，以及发出者（只有两个值，“个人用户”或者“社团管理员”）。所有参与互动的个人用户和社团管理员都会产生相应的聊天记录。这个关系集中的个人用户和社团管理员是部分参与，且是多对多的关系，因为并非所有的个人用户和社团管理员都会产生聊天行为。

2. 个人用户和活动的**浏览记录**关系：

当个人用户在平台上查看活动时，系统会在浏览记录表中生成相应的记录，它的描述性属性包括：浏览起始时间、是否报名。只有进行浏览行为的个人用户和被浏览的活动会产生浏览记录。这个关系集中的个人用户和活动是部分参与，且是多对多的关系，因为并非所有的个人用户都会浏览活动，同样并非所有的活动都会被个人用户浏览。

3. 活动和社团管理员的**管理**关系：

在这个关系中，活动是完全参与，因为每个活动都必须由某个社团管理员进行管理；而社团管理员是部分参与，因为并非所有的社团管理员都会管理活动。这个关系集是多对一的关系，因为一个社团管理员可能管理多个活动，但每个活动只能由一个社团管理员进行管理。

4. 社团管理员和公告的**发布**关系：

当社团管理员在平台上发布公告时，系统会在记录公告的发布社团管理员的ID。在这个关系中，公告是完全参与，因为每个公告都必须由某个社团管理员进行发布；而社团管理员是部分参与，因为并非所有的社团管理员都会发布公告。这个关系集是一对多的关系，因为一个社团管理员可能发布多个公告，但每个公告只能由一个社团管理员进行发布。

5. 个人用户和活动的**收藏**关系：

当个人用户在平台上收藏活动时，系统会在收藏记录表中生成相应的记录。它的描述性属性包括收藏的时间戳。这个关系集是多对多的关系，因为一个个人用户可能收藏多个活动，同样一个活动也可能被多个个人用户收藏。

6. 个人用户和订单的**下单**关系：

当个人用户在平台上报名活动后，系统会生成相应的订单，并在订单记录表中生成记录。在这个关系中，订单是完全参与，因为每个订单都必须由某个个人用户下单；而个人用户是部分参与，因为并非所有的个人用户都会下单。这个关系集是一对多的关系，因为一个个人用户可能下多个订单，但每个订单只能由一个个人用户下单。

7. 订单和活动的**报名**关系：

活动被报名，生成订单。这是一个多对一关系，订单完全参与，活动部分参与。

8. 用户和评论的**发布**关系：

当用户在平台上发表评论时，系统会在评论记录表中生成相应的记录。在这个关系中，评论是完全参与，因为每条评论都必须由某个用户发布；而用户是部分参与，因为并非所有的用户都会发表评论。这个关系集是一对多的关系，因为一个用户可能发表多条评论，但每条评论只能由一个用户发布。

9. 活动和评论的**反馈**关系：

当用户对活动发表评论时，系统会在评论记录表中生成相应的记录，并将该评论与对应的活动关联起来。在这个关系中，评论是完全参与，因为每条评论都必须针对某个活动；而活动是部分参与，因为并非所有的活动都会收到评论。这个关系集是一对多的关系，因为一个活动可能收到多条评论，但每条评论只能针对一个活动。

10. 评论的**父子**关系：

在平台上，用户不仅可以对活动发表评论，还可以对其他评论发表评论，形成评论的层级结构，这是一种递归的关系。系统会在评论记录表中生成相应的记录，并将每条评论与其对应的父评论和子评论关联起来。这种关系模型构成了一个层次结构，从最初对活动的评论（没有父评论的评论），逐级展开到对评论的评论，最终形成活动的评论区。

11. 评论和申诉的**被投诉**关系：

当用户或者社团管理员提出申诉时，系统会在申诉记录表中生成相应的记录，并将申诉与被投诉的评论关联起来。在这个关系中，申诉和评论是一对零或一的关系，因为每次申诉只能针对一条评论，也可能没有评论被投诉；同样，评论和申诉也是一对零或一的关系，因为一条评论可能被申诉，也可能没有被申诉。这个关系集表明，每次申诉与被投诉的评论存在着唯一性的对应关系。

12. 活动和申诉的**被投诉**关系：

当用户对某个活动提出申诉时，系统会在申诉记录表中生成相应的记录，并将申诉与被投诉的活动关联起来。在这个关系中，申诉是一对零或一的关系，因为每次申诉只能针对一个活动，也可能没有活动被投诉；同样，活动也是一对零或一的关系，因为一个活动可能被申诉，也可能没有被申诉。这个关系集表明，每次申诉与被投诉的活动存在着唯一性的对应关系。

13. 申诉和用户的**被投诉**关系：

当社团管理员对另一位用户（例如该用户的评论或行为）提出申诉时，系统会在申诉记录表中生成相应的记录，并将申诉与被投诉的用户关联起来。在这个关系中，申诉是一对零或一的关系，因为每次申诉只能针对一个用户，也可能没有用户被投诉；同样，用户也是一对零或一的关系，因为一个用户可能被申诉，也可能没有被申诉。这个关系集表明，每次申诉与被投诉的用户存在着唯一性的对应关系。

14. 用户和申诉的**提出申诉**关系：

用户在发现问题或不满时，可以向系统提出申诉。系统会在申诉记录表中生成相应的记录，并将申诉与提出申诉的用户关联起来。在这个关系中，申诉是完全参与，因为每次申诉都必须由一个用户提出；而用户是部分参与，因为并非所有的用户都会提出申诉。这个关系集是多对一的关系，因为一个用户可能提出多次申诉，但每次申诉只能由一个用户提出。

2.5.1.3 数据库E-R图设计

暂无内容

2.5.2 数据库物理设计

关系模型的物理结构实质上是一系列关系模式的集合。为了构建数据库的物理结构，我们需要将E-R图中定义的实体、属性及实体间关系转化为具体的关系模式。我们遵循以下转换原则：

- 实体类型的转换：每个实体集对应一个关系模式，实体属性成为关系属性，实体码成为关系码。
- 一对一关系：可转换为独立的关系模式，或与任一端的关系模式合并。作为独立关系模式时，涉及的实体码及关系属性均转化为关系属性，各实体码作为候选码。合并时，需在对应关系模式中加入另一端的码及关系属性。
- 一对多关系：可转换为独立的关系模式，或与多端的关系模式合并。作为独立关系模式时，涉及的实体码及关系属性均转化为关系属性，而关系码为多端实体的码。
- 多对多关系：转换为一个关系模式，涉及的实体码组合成为关系码。

- 三元或更多元关系：转换为一个关系模式，涉及的实体码及关系属性均转化为关系属性，关系码为实体码的组合。
- 具有相同键的关系模式可进行合并。
  依照上述原则，我们将E-R图转换为以下数据库结构，并以表格形式详细列出了数据库表的名称、标识符、数据项、记录、存储要求、存储类型、索引、存储区域等信息：

1. **<table>用户user表**

字段名	存储类型	存储要求	索引	描述	备注
id	int	**PK**		用户ID	
username	varchar	Not null	unique	用户名	社团管理员为社团编号；普通用户为学号；系统管理员为"admin"+三位数字
password	varchar	Not null		密码	表中存储的是实际密码的md5值*
email	varchar	Not null	unique	邮箱	
phone	varchar	Not null	unique	电话号码	
campus	enum	Not null		所在校区	0：四平路校区/1：嘉定校区/2：沪西校区/3：沪北校区/4：其它校区
login_status	int	Not null		登陆状态	0：未登录/1：登录
account_status	int	Not null		账号状态	0：封禁/1：正常/2：待审核
balance	decimal	Not null		账户余额	
pay_password	varchar	Not null		支付提现密码	
reg_time	datetime	Not null		注册时间	
role	int			用户类型	

		Not null			0：个人用户/1：社团管理员/2：系统管理员

2. <table>社团活动关键词keywords表

字段名	存储类型	存储要求	索引	描述	备注
keyword	varchar	**PK**		活动类型关键词	

3. <table>社团管理员society表

字段名	存储类型	存储要求	索引	描述	备注
soc_id	int	**PK，FK**		账号ID	**引用自user表**
soc_name	varchar	Not null	unique	社团名称	
soc_intro	text	Not null		社团介绍	
soc_type	enum	Not null		社团类别	
soc_logo	varchar	Not null		社团头像图片	表中存储图片相对路径*

4. <table>社团负责人society_admin表

字段名	存储类型	存储要求	索引	描述	备注
soc_id	int	**PK，FK**		账号ID	**引用自society表**
soc_admin_no	varchar	**PK**		负责人学号	

soc_admin_name	varchar	Not null		负责人姓名	
soc_admin_phone	varchar	Not null		负责人电话号码	
soc_admin_email	varchar	Not null		负责人邮箱	

5. <table>社团关键词society_keyword表

字段名	存储类型	存储要求	索引	描述	备注
soc_id	int	**PK，FK**		账号ID	**引用自society表**
keyword	varchar	**PK，FK**		社团关键词	引用自keywords表

6. <table>社团图片society_image表

字段名	存储类型	存储要求	索引	描述	备注
soc_id	int	PK，FK		社团ID	
soc_image	varchar	PK		社团图片	以相对路径的方式存储

7. <table>学生student表

字段名	存储类型	存储要求	索引	描述	备注
stu_id	int	PK，FK		账号ID	引用自user表
stu_name	varchar	Not null		学生姓名	
stu_year	enum				

		Not null		学生所在年级	
stu_school	enum	Not null		学生所在学院	
stu_major	varchar	Not null		学生所在专业	
stu_notes	text	Not null		个人备注	由个人设置*

8. <table>学生喜好关键词student_keyword表

字段名	存储类型	存储要求	索引	描述	备注
stu_id	int	PK，FK		账号ID	引用自student表
keyword	varchar	PK，FK		活动类别	引用自keywords表
prefer_weight	decimal			喜好标签权重	推荐算法确定*

9. <table>社团活动activity表

字段名	存储类型	存储要求	索引	描述	备注
act_id	int	PK		活动ID	
act_name	varchar	Not null		活动名称	管理员可修改，管理员备注内容融合在这里*
act_intro	text	Not null		活动简介	社团、管理员都可以修改*
act_location	varchar	Not null		活动位置	
ticket_price					

	decimal	Not null		活动报名费	
upload_time	datetime	Not null		活动发布时间	
reg_start_time	datetime	Not null		活动开始报名时间	应大于发布时间
reg_end_time	datetime	Not null		活动截止报名时间	应大于活动开始报名时间
act_time	datetime	Not null		活动时间	
act_capacity	int	Not null		活动总容量	应大于等于0
act_left	int	Not null		活动剩余名额	应大于等于0
act_rating	decimal	Not null		活动评分	
rating_num	int	Not null		评分人数	
soc_id	int	FK		活动发布社团id	引用自society表

10. <table>活动关键词activity_keyword表

字段名	存储类型	存储要求	索引	描述	备注
act_id	int	PK，FK		活动ID	引用自activity表
keyword	varchar	FK		活动关键词	引用自keywords表

11. <table>活动图片activity_image表

字段名	存储类型	存储要求	索引	描述	备注

act_id	int	PK，FK		活动ID	外键依赖activity的act_ID
act_image	varchar	PK		活动图片	以相对路径方式存储

12. <table>活动报名订单indent表

字段名	存储类型	存储要求	索引	描述	备注
ind_id	int	PK		订单ID	
ind_price	decimal	Not null		支付金额	
ind_create_time	datetime	Not null		创建时间	
ind_verify_code	varchar2	Not null		活动参与核销码	
ind_name	varchar	Not null		报名者姓名	
ind_stu_no	varchar	Not null		报名者学号	
ind_notes	text			用户备注	
ind_status	int	Not null		订单状态	0：待核销/1：已完成/2：已退款
ind_rating	decimal			评分	
act_id	int	FK		活动ID	引用自activity表
stu_id	int	FK		学生账号ID	引用自student表
soc_id	int	FK		社团账号ID	引用自society表
ind_rtime	datetime			退款时间	可空

ind_rnotes	text			退款原因	可空
ind_rmoney	decimal			退款金额	可空

13. <table>评论comment表

字段名	存储类型	存储要求	索引	描述	备注
cmt_id	int	PK		评论ID	
cmt_father	int	Not null		父评论	0：无父评论/非0：父评论ID*
cmt_content	text	Not null		评论内容	
cmt_time	datetime	Not null		评论时间	
act_id	int	FK		活动ID	引用自 activity表
user_id	int	FK		用户ID	引用自 user表

14. <table>申诉appeal表

字段名	存储类型	存储要求	索引	描述	备注
app_id	int	PK		申诉ID	
app_time	datetime	Not null		申诉时间	
app_matters	int	Not null		申诉事项	1：评论/2：活动/3：社团/4：个人用户
app_content	text	Not null		申诉描述	
user_id	int	FK		申诉用户ID	可空，引用自 user表
act_id	int	FK		申诉活动ID	可空，引用自 activity表

cmt_id	int	FK		申诉评论ID	可空，引用自 comment表
complainant_id	int	FK		发起申诉用户ID	引用自 user表

15. <table>申诉图片appeal_image表

字段名	存储类型	存储要求	索引	描述	备注
app_id	int	PK,FK		申诉ID	外键依赖appeal的app_ID
app_image	varchar	PK		截图图像	以相对路径方式存储

16. <table>公告notice表

字段名	存储类型	存储要求	索引	描述	备注
ntc_id	int	PK		公告ID	
ntc_time	datetime			发布时间	
ntc_content	text			公告内容	
ntc_status	int			显示状态	0：未显示/1：已显示
soc_id	int	FK		社团ID	引用自 society表

17. <table>收藏favour表

字段名	存储类型	存储要求	索引	描述	备注
act_id	int	PK，FK		活动ID	引用了activity表中的act_id属性
stu_id	int	PK，FK		收藏者	引用了student表中的stu_id属性

18. <table>浏览记录browse表

字段名	存储类型	存储要求	索引	描述	备注
bro_time_start	TIMESTAMP(3)	PK		起始浏览时间	非空
act_id	int	PK，FK		活动ID	引用了activity表中的act_id属性
browser_id	int	PK，FK		浏览者ID	引用了student表中的stu_id属性
whether_buy	int	Not null		报名情况	0:未报名/1:报名

19. **<table>聊天记录chat表**

字段名	存储类型	存储要求	索引	描述	备注
chat_time	TIMESTAMP(3)	PK		聊天时间	非空
stu_id	int	PK，FK		个人用户ID	引用了student表中的stu_id属性
soc_id	int	PK，FK		社团ID	引用了society表中的soc_id属性
chat_content	text	Not null		聊天内容	
chat_sender	bool	Not null		发出者	0：个人用户/1：社团

20. <table>系统管理员admin表

字段名	存储类型	存储要求	索引	描述	备注

21. <table>活动场地place表

字段名	存储类型	存储要求	索引	描述	备注

22. <table>礼券coupon表

字段名	存储类型	存储要求	索引	描述	备注

2.5.3 数据库设计考虑

2.5.3.1 数据库范式的应用

考虑到系统涉及多个数据表及其频繁的数据操作（增删改查），遵循一定的数据库范式至关重要，以防止操作异常，减少数据冗余。我们的关系模式满足以下数据库范式：

- 第一范式（1NF）：关系模式中所有域均为原子性，数据表中的每一项都是不可分割的单元。
- 第二范式（2NF）：每个实例或记录通过唯一标识符进行区分。
- 第三范式（3NF）：关系中的每个元素不包含在其他关系中出现的非主键信息，任何非主属性都不应传递依赖于主键。

2.5.3.2 查询存储的优化策略

随着系统运行时间的增长，数据库中数据量增大，负载加重，查询效率对系统响应时间的影响日益明显。为此，我们对数据库表建立了必要的索引，以提升查询速度。我们遵循以下原则建立简单索引和组合索引：

- 表的外键默认建立索引。
- 针对查询中常出现的条件字段建立索引。
- 非关键字段尽量不建立索引，减少存储负担。此外，我们还针对一些复杂查询创建视图，以简化查询过程，便于管理。

2.5.3.3 函数和触发器

鉴于系统涉及多项紧密关联的业务逻辑，需要实时更新（例如，用户积分达到一定水平时自动升级，用户在特定时间内获得礼券，用户未评价时系统自动默认好评，订单状态的实时维护等），这些逻辑在后端难以实现。因此，我们设计了相应的函数和触发器，以自动维护这些功能。

2.6 系统出错处理设计