# Bit-Database

## 表结构

### Institution 模型

#### 描述

Institution 模型代表研究所的详细信息，包括名称、简介、联系方式等。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 | 默认值 | 关联模型 |
| --- | --- | --- | --- | --- | --- |
| id | Integer | 研究所 ID，主键 | 否 | 自增 |  |
| name | String(255) | 研究所名称 | 否 |  |  |
| short\_name | String(255) | 研究所简称 | 否 |  |  |
| introduction | Text | 研究所简介 | 否 |  |  |
| address | String(255) | 研究所地址 | 否 |  |  |
| phone | String(20) | 研究所电话 | 否 |  |  |
| fax | String(20) | 研究所传真 | 否 |  |  |
| email | String(255) | 研究所邮箱 | 否 |  |  |
| website | String(255) | 研究所网站 | 否 |  |  |
| logo\_path | String(255) | 研究所 logo 路径 | 否 |  |  |
| created\_at | DateTime | 创建时间 | 是 | 当前时间戳 |  |
| is\_active | Boolean | 是否启用 | 是 | False |  |
| operator\_id | String(20) | 更新人 ID，外键 | 否 |  | Users |
| operator | Object | 更新人对象，反向引用 | 是 |  | Users |

#### 反向引用

* operator: 通过 Users 模型关联更新人的详细信息。

#### 默认值

* 新建研究所记录时，is\_active 默认为 False，表示未启用。
* created\_at 默认为当前时间戳。

#### 外键关系

* operator\_id 字段作为外键与 Users 表的 work\_id 字段关联。

### Users 模型

#### 描述

Users 模型代表系统中的用户账户，包含用户的认证信息和个人资料的外键。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 | 默认值 | 关联模型 |
| --- | --- | --- | --- | --- | --- |
| id | Integer | 用户 ID，主键 | 否 | 自增 |  |
| work\_id | String(20) | 工号 | 否 |  |  |
| password\_hash | String(528) | 密码哈希 | 否 |  |  |
| created\_at | DateTime | 创建时间 | 是 | 当前时间戳 |  |
| last\_login | DateTime | 最后登录时间 | 是 | None |  |
| login\_fail\_count | Integer | 登录失败次数 | 是 | 0 |  |
| last\_fail\_login | DateTime | 上次登录失败时间 | 是 | None |  |
| is\_active | Boolean | 是否激活 | 是 | False |  |
| user\_info\_id | Integer | 用户信息 ID，外键 | 否 |  | UserInfo |
| user\_info | Object | 用户信息对象，反向引用 | 是 |  | UserInfo |
| library\_status\_id | Integer | 图书馆常量设置 ID，外键 | 否 |  | LibraryStatus |
| library\_status | Object | 图书馆常量设置对象，反向引用 | 是 |  | LibraryStatus |

#### 方法说明

* get\_english\_name(self): 返回用户信息中的英文名字段。
* get\_teacher(self, zheng, type): 根据提供的 zheng 和 type 参数，返回学生的导师或共同导师的名称或 ID。
* @property username: 返回用户信息中的用户名字段。
* set\_password(self, password): 设置用户的密码，通过哈希处理。
* check\_password(self, password): 检查提供的密码是否与用户的哈希密码匹配。
* get\_id(self): 返回用户 ID 的字符串表示。
* activate(self): 激活用户账户。
* reset\_password(self, new\_password): 重置用户密码为提供的新密码。
* update\_info(self, \*\*kwargs): 根据提供的关键字参数更新用户信息。
* to\_dict(self): 返回用户的字典表示，不包括 user\_info 和 library\_status 字段。

#### 反向引用

* user\_info: 通过 UserInfo 模型关联用户详细信息。
* library\_status: 通过 LibraryStatus 模型关联图书馆状态信息。

#### 默认值

* 新建用户时，is\_active 默认为 False，表示未激活。
* last\_login 默认为 None。
* login\_fail\_count 默认为 0。
* password\_hash 默认为哈希处理后的 '123456'。

### UserInfo 模型

#### 描述

UserInfo 模型存储用户详细信息，包括用户名、联系方式、性别、用户类型等。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 | 默认值 | 备注 |
| --- | --- | --- | --- | --- | --- |
| id | Integer | 用户信息 ID，主键 | 否 | 自增 |  |
| email | String(100) | 邮箱 | 是 |  |  |
| phone | String(20) | 手机号 | 是 |  |  |
| username | String(50) | 用户名 | 否 |  |  |
| english\_name | String(50) | 英文姓名 | 是 | 拼音转换自username（若未提供） |  |
| gender | String(10) | 性别 | 是 |  |  |
| user\_type | String(20) | 用户类型 | 否 |  |  |
| photo\_path | String(255) | 照片路径 | 是 | “images/avatar.png” | 默认头像路径 |
| birth\_date | Date | 出生日期 | 是 |  |  |
| nationality | String(50) | 国籍 | 是 |  |  |
| remarks | Text | 备注信息 | 是 |  |  |

#### 方法说明

* \_\_init\_\_(self, user\_id, username, phone, user\_type, gender, email=None, nationality=None, birth\_date=None, english\_name=None, remarks=None, photo\_path="images/avatar.png"): 构造函数，初始化用户信息。若未提供邮箱，则以user\_id加”@bit.edu.cn“作为默认邮箱；若未提供英文姓名，则使用中文姓名的拼音。
* to\_dict(self): 返回用户的字典表示。

#### 默认值

* photo\_path 默认为 “images/avatar.png”，表示默认头像路径。
* 邮箱默认为 user\_id 加 “@bit.edu.cn”，如果提供了邮箱则使用提供的邮箱。

#### 备注

* 英文姓名如果没有提供，则自动使用中文姓名的拼音作为英文姓名。

#### 图书馆常量设置表：（LibraryStatus）

* id: 图书馆常量设置 ID，主键
* user\_id: 用户 ID，外键关联 Users 表
* interval\_date: 间隔日期 天数 默认 5
* borrow\_period: 借阅期限 天数 默认 30
* overdue\_reminder\_days: 超出期限提前提醒天数 默认 3
* borrow\_limit: 单个用户借阅数量限制 默认 2
* violation\_limit: 单个用户违规记录数量限制 默认 3
* is\_book\_admin: 是否图书管理员

### Students 模型

#### 描述

Students 模型代表学生信息，包括学生类别、入学时间、导师信息等，与Users模型通过user\_id关联。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 | 默认值 |
| --- | --- | --- | --- | --- |
| id | Integer | 学生 ID，主键 | 否 | 自增 |
| user\_id | String(20) | 关联的用户 ID，外键 | 否 |  |
| category | String(20) | 学生类别（本科、硕士、博士） | 否 |  |
| admission\_time | Date | 入学时间 | 否 |  |
| user | Object | 关联的用户对象，反向引用 | 是 |  |
| tutor\_id | String(50) | 导师 ID | 是 |  |
| co\_tutor\_id | String(50) | 副导师 ID | 是 |  |
| tutor\_name | String(50) | 导师姓名 | 是 |  |
| co\_tutor\_name | String(50) | 副导师姓名 | 是 |  |
| graduation\_time | Date | 毕业时间 | 是 |  |
| first\_employment\_unit | String(50) | 初次就业单位 | 是 |  |

#### 构造函数说明

* \_\_init\_\_(self, student\_id, category, admission\_time, username, phone, gender, user\_type, tutor\_id=None, co\_tutor\_id=None, tutor\_name=None, co\_tutor\_name=None, graduation\_time=None, first\_employment\_unit=None, email=None, nationality=None, birth\_date=None, remarks=None, english\_name=None): 初始化学生信息。如果未提供tutor\_id或co\_tutor\_id，则字段设置为None。同时创建关联的Users记录。

#### 反向引用

* user: 通过Users模型关联用户信息。

#### 默认值

* 无特定字段默认值，但tutor\_id、co\_tutor\_id、tutor\_name、co\_tutor\_name、graduation\_time和first\_employment\_unit在未提供时将被设置为None。

#### 外键关系

* user\_id字段作为外键与Users表的work\_id字段关联。

### Teachers 模型

#### 描述

Teachers 模型代表教师信息，包括职称、学历等，与Users模型通过user\_id唯一关联。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 | 默认值 |
| --- | --- | --- | --- | --- |
| id | Integer | 教师 ID，主键 | 否 | 自增 |
| title | String(50) | 职称 | 是 |  |
| education | String(50) | 教师学历 | 是 |  |
| user\_id | String(20) | 关联的用户 ID，外键 | 否 |  |
| user | Object | 关联的用户对象 | 是 |  |

#### 构造函数说明

* \_\_init\_\_(self, title, teacher\_id, username, phone, gender, user\_type, education, email=None, nationality=None, birth\_date=None, english\_name=None, remarks=None): 初始化教师记录，包括职称、学历等信息，并创建一个与之关联的Users对象。

#### 反向引用

* user: 通过Users模型关联用户详细信息。

#### 默认值

* 无特定字段默认值，但title与education在未提供时将被设置为None。

#### 外键关系

* user\_id字段作为外键与Users表的work\_id字段关联，并具有唯一性约束。

#### 唯一性约束

* user\_id: 必须唯一，确保每个教师的用户 ID 在数据库中只存在一次。

### PartTimeTeachers 模型

#### 描述

PartTimeTeachers 模型代表兼职教师的详细信息，与Teachers模型通过teacher\_id一对一关联。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 | 默认值 |
| --- | --- | --- | --- | --- |
| id | Integer | 兼职教师 ID，主键 | 否 | 自增 |
| work\_unit | String(50) | 工作单位 | 是 |  |
| teacher\_id | Integer | 教师工号，外键 | 否 |  |
| teacher | Object | 关联的教师实体 | 是 |  |

#### 构造函数说明

* \_\_init\_\_(self, work\_unit, teacher\_id, title, username, phone, gender, education, email=None, nationality=None, birth\_date=None, english\_name=None, remarks=None): 初始化兼职教师记录，包括工作单位等信息，并创建一个与之关联的Teachers对象。

#### 反向引用

* teacher: 通过Teachers模型关联教师详细信息。

#### 默认值

* 无特定字段默认值，但work\_unit在未提供时将被设置为None。

#### 外键关系

* teacher\_id字段作为外键与Teachers表的id字段关联。

#### 示例构造函数调用

创建一个兼职教师记录时，您需要提供工作单位、教师工号、职称、用户名、手机号、性别、学历等信息，其他字段如邮箱、国籍、出生日期、英文姓名和备注信息是可选的。兼职教师的用户类型默认设置为’兼职教师’。

### FullTimeTeachers 模型

#### 描述

FullTimeTeachers 模型代表全职教师的详细信息，与Teachers模型通过teacher\_id一对一关联。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 | 默认值 |
| --- | --- | --- | --- | --- |
| id | Integer | 全职教师 ID，主键 | 否 | 自增 |
| qualification | String(50) | 导师资格 | 是 |  |
| duty | Text | 研究所职务 | 是 |  |
| social\_part\_time | Text | 社会兼职 | 是 |  |
| administrative\_duty | Text | 学院行政职务 | 是 |  |
| office\_phone | String(20) | 办公电话 | 是 |  |
| teacher\_id | Integer | 教师工号，外键 | 否 |  |
| teacher | Object | 关联的教师实体 | 是 |  |

#### 构造函数说明

* \_\_init\_\_(self, teacher\_id, title, qualification, duty, social\_part\_time, administrative\_duty, office\_phone, username, phone, gender, education, email=None, nationality=None, birth\_date=None, english\_name=None, remarks=None): 初始化全职教师记录，包括导师资格、研究所职务等信息，并创建一个与之关联的Teachers对象。

#### 反向引用

* teacher: 通过Teachers模型关联教师详细信息。

#### 默认值

* office\_phone: 如果未提供，则默认为空字符串。

#### 外键关系

* teacher\_id字段作为外键与Teachers表的id字段关联。

#### 用户类型

* 教师的用户类型默认设置为’全职教师’。

### LoginEvent 模型

#### 描述

LoginEvent 模型用于记录用户登录系统的相关事件信息。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 | 默认值 |
| --- | --- | --- | --- | --- |
| id | Integer | 登录事件 ID，主键 | 否 | 自增 |
| user\_id | String(20) | 用户工号，外键关联到 Users 表 | 是 |  |
| user\_name | String(50) | 用户名称 | 是 |  |
| ip\_address | String(45) | 登录 IP 地址 | 是 |  |
| login\_time | DateTime | 登录时间 | 是 | 当前时间戳 (datetime.now) |
| logout\_time | DateTime | 登出时间，用户会话结束时间 | 是 |  |
| session\_id | String(255) | 会话 ID，用于跟踪用户会话 | 是 |  |
| tutor\_id | String(50) | 导师 ID，如果用户是导师 | 是 |  |
| co\_tutor\_id | String(50) | 副导师 ID，如果用户是副导师 | 是 |  |

#### 反向引用

* user: 通过Users模型关联用户详细信息，允许从登录事件查询到用户信息。

#### 默认值

* login\_time: 默认设置为事件发生时的时间戳。

#### 外键关系

* user\_id: 与Users表的work\_id字段相关联，标识登录的用户。

#### 可空字段说明

* logout\_time: 对于仍在进行中的会话，该字段可能为NULL。
* tutor\_id和co\_tutor\_id: 如果用户不是导师或副导师，这些字段可能为NULL。

### TeachingWorksTeacherAssociation 模型

#### 描述

TeachingWorksTeacherAssociation 模型用于关联教学作品与教师，确保每个项目与参与的教师之间有明确的联系。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 关联记录 ID，主键 | 否 |
| project\_id | Integer | 教学作品 ID，外键 | 否 |
| user\_id | String(20) | 用户工号，外键关联到 Users 表 | 否 |

#### 外键关系

* project\_id: 与TeachingWork表的id字段相关联，标识教学作品。
* user\_id: 与Users表的work\_id字段相关联，标识参与教学作品的教师。

#### 模型功能

* 该模型允许将特定的教学作品与教师的工作编号相关联，确保教学责任和成果的归属明确。

#### 约束条件

* project\_id和user\_id字段由于是外键，因此不能包含空值，确保每条记录都有明确的教学作品和教师关联。

### TeachingWork 模型

#### 描述

TeachingWork 模型代表教学作品的详细信息，包括课程编号、名称、性质、学生层次和授课时间。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 课程 ID，主键 | 否 |
| course\_id | String(20) | 课程编号 | 否 |
| course\_name | String(100) | 课程名称 | 否 |
| course\_nature | String(50) | 课程性质 | 否 |
| student\_level | String(50) | 学生层次 | 否 |
| teaching\_time | String(50) | 授课时间 | 否 |

#### 关联模型

* owner: 定义了与Users模型的多对多关系，通过TeachingWorksTeacherAssociation表进行关联。

#### 构造函数说明

* \_\_init\_\_(self, course\_id, course\_name, course\_nature, student\_level, teaching\_time): 构造函数用于初始化一个新的TeachingWork实例，需要提供课程编号、名称、性质、学生层次和授课时间。

#### 多对多关系

* 教学作品与教师之间存在多对多关系，即一个教学作品可以有多个教师参与，一个教师也可以参与多个教学作品。

#### 默认值

* 无默认值，所有字段在创建记录时都需要明确指定。

#### 索引建议

* course\_id: 作为课程编号，建议设置为唯一且可索引，以优化查询效率。

### ResearchWorkTeacherAssociation 模型

#### 描述

ResearchWorkTeacherAssociation 模型用于建立科研项目与教师之间的关联关系，支持多对多的关联方式。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 关联记录 ID，主键 | 否 |
| project\_id | Integer | 科研项目 ID，外键 | 否 |
| user\_id | String(20) | 用户工号，外键关联到 Users 表 | 否 |

#### 外键关系

* project\_id: 与ResearchWork表的id字段相关联，标识科研项目。
* user\_id: 与Users表的work\_id字段相关联，标识参与科研项目的教师。

#### 模型功能

* 该模型允许将特定的科研项目与教师的工作编号相关联，明确科研项目的参与人员。

#### 约束条件

* project\_id和user\_id字段由于是外键，因此不能包含空值，确保每条记录都有明确的科研项目和教师关联。

### ResearchWork 模型

#### 描述

ResearchWork 模型存储科研工作的详细信息，包括项目编号、名称、性质、开始和结束日期。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 科研工作 ID，主键 | 否 |
| project\_id | String(20) | 项目编号 | 否 |
| project\_name | String(100) | 项目名称 | 否 |
| project\_nature | String(50) | 项目性质 | 否 |
| start\_date | Date | 项目开始日期 | 是 |
| end\_date | Date | 项目结束日期 | 是 |

#### 关联模型

* owner: 定义了与Users模型的多对多关系，通过ResearchWorkTeacherAssociation表进行关联。

#### 构造函数说明

* \_\_init\_\_(self, project\_id, project\_name, project\_nature, start\_date, end\_date): 构造函数用于初始化一个新的ResearchWork实例，需要提供项目编号、名称、性质、开始和结束日期。

#### 多对多关系

* 科研工作与教师之间存在多对多关系，即一个科研工作可以有多个教师参与，一个教师也可以参与多个科研工作。

#### 默认值

* start\_date和end\_date字段可以为空，允许记录尚未开始或持续进行中的项目。

#### 索引建议

* project\_id: 作为项目编号，建议设置为唯一且可索引，以优化查询效率。

#### 描述

研究工作表用于存储科研工作的详细信息，包括项目编号、项目名称、项目性质、项目开始和结束日期。此外，通过多对多关系表ResearchWorkTeacherAssociation与用户表关联，以表示哪些教师参与了哪些科研工作。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 | 默认值 | 关联模型 |
| --- | --- | --- | --- | --- | --- |
| **id** | Integer | 科研工作 ID，主键 | 否 | 自增 |  |
| **project\_id** | String(20) | 项目编号 | 否 | None |  |
| **project\_name** | String(100) | 项目名称 | 否 | None |  |
| **project\_nature** | String(50) | 项目性质 | 否 | None |  |
| **start\_date** | Date | 项目开始日期 | 是 | None |  |
| **end\_date** | Date | 项目结束日期 | 是 | None |  |

#### 多对多关系

* owner: 通过ResearchWorkTeacherAssociation表与Users表建立多对多关系，表示科研工作和教师之间的关联。

#### 初始化方法

* \_\_init\_\_(self, project\_id, project\_name, project\_nature, start\_date, end\_date, teachers): 初始化科研工作记录，并将教师与科研工作相关联。

#### 方法说明

* create\_teachers\_association(self, teachers): 创建科研工作与教师之间的关联记录。

#### 默认值

* start\_date 和 end\_date: 项目开始和结束日期字段可以为空，允许记录尚未开始或结束日期未知的项目。

#### 反向引用

* research\_works: 在Users模型中，通过

### TeachingAchievementTeacherAssociation 模型

#### 描述

TeachingAchievementTeacherAssociation 模型用于关联教学成果与教师，实现多对多的关系。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 关联记录 ID，主键 | 否 |
| user\_id | String(20) | 用户工号，外键 | 否 |
| project\_id | Integer | 教学成果 ID，外键 | 否 |

#### 外键关系

* user\_id: 与Users表的work\_id字段相关联，标识与教学成果关联的教师。
* project\_id: 与TeachingAchievements表的id字段相关联，标识具体的教学成果。

#### 模型功能

* 该模型允许将教师与其参与的教学成果进行关联，支持一个教师关联多个教学成果，以及一个教学成果被多个教师共同参与。

#### 约束条件

* user\_id和project\_id字段由于是外键，因此不能为空，确保每条记录都有明确的教师和教学成果的关联。

### TeachingAchievements 模型

#### 描述

TeachingAchievements 模型用于存储教学成果的详细信息，包括成果名称、级别、年份和描述。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 教学成果 ID，主键 | 否 |
| achievement\_year | String(50) | 成果年份 | 否 |
| name | String(255) | 成果名称 | 否 |
| level | String(50) | 级别 | 否 |
| description | Text | 成果描述 | 是 |

#### 关联模型

* owner: 定义了与Users模型的多对多关系，通过TeachingAchievementTeacherAssociation表进行关联，表示拥有该教学成果的教师。

#### 构造函数说明

* \_\_init\_\_(self, level, achievement\_year, name, description): 构造函数用于初始化一个新的TeachingAchievements实例，需要提供成果的级别、年份、名称和描述。

#### 多对多关系

* 教学成果与教师之间存在多对多关系，即一个教学成果可以有多个教师参与，一个教师也可以拥有多个教学成果。

#### 默认值

* 无默认值，所有字段在创建记录时都需要明确指定。

#### 索引建议

* achievement\_year和name: 可以根据查询需求考虑设置为索引，以优化基于成果年份或名称的查询效率。

### PaperAuthorAssociation 模型

#### 描述

PaperAuthorAssociation 模型用于建立论文与其作者之间的关联关系，支持一对多的关联方式。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 关联记录 ID，主键 | 否 |
| project\_id | Integer | 论文 ID，外键 | 否 |
| user\_id | String(20) | 用户工号，外键 | 否 |

#### 外键关系

* project\_id: 与Papers表的id字段相关联，标识具体的论文。
* user\_id: 与Users表的work\_id字段相关联，标识论文的作者。

#### 模型功能

* 该模型允许将论文与其作者进行关联，支持一篇论文有多个作者，以及一个作者可以撰写多篇论文。

#### 约束条件

* project\_id和user\_id字段由于是外键，因此不能为空，确保每条记录都有明确的论文和作者的关联。

### Papers 模型

#### 描述

Papers 模型用于存储论文的详细信息，包括发表年份、类型、题目、期刊/会议名称和刊号/会议时间。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 论文 ID，主键 | 否 |
| publication\_year | String(50) | 发表年份 | 否 |
| type | String(50) | 类型：教学，科研 | 否 |
| title | String(255) | 论文题目 | 否 |
| publication | String(100) | 期刊/会议名称 | 否 |
| publication\_number | String(100) | 刊号/会议时间 | 否 |

#### 关联模型

* authors: 定义了与Users模型的多对多关系，通过PaperAuthorAssociation表进行关联，表示论文的作者。

#### 构造函数说明

* \_\_init\_\_(self, publication\_year, type, title, publication, publication\_number): 构造函数用于初始化一个新的Papers实例，需要提供论文的发表年份、类型、题目、期刊/会议名称和刊号/会议时间。

#### 多对多关系

* 论文与作者之间存在多对多关系，即一篇论文可以有多个作者，一个作者也可以撰写多篇论文。

#### 默认值

* 无默认值，所有字段在创建记录时都需要明确指定。

#### 索引建议

* publication\_year和title: 可以根据查询需求考虑设置为索引，以优化基于发表年份或论文题目的查询效率。

### BookAuthorAssociation 模型

#### 描述

BookAuthorAssociation 模型用于建立教科书或书籍与其作者之间的关联关系，支持一对多的关联方式。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 关联记录 ID，主键 | 否 |
| project\_id | Integer | 书籍 ID，外键 | 否 |
| user\_id | String(20) | 用户工号，外键 | 否 |

#### 外键关系

* project\_id: 与Textbooks表的id字段相关联，标识具体的书籍或教科书。
* user\_id: 与Users表的work\_id字段相关联，标识书籍的作者。

#### 模型功能

* 该模型允许将书籍与其作者进行关联，支持一本书有多个作者，以及一个作者可以撰写多本书。

#### 约束条件

* project\_id和user\_id字段由于是外键，因此不能为空，确保每条记录都有明确的书籍和作者的关联。

### TextBooks 模型

#### 描述

TextBooks 模型用于存储教材的详细信息，包括出版年份、名称和获奖信息。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 教材 ID，主键 | 否 |
| publication\_year | String(50) | 出版年份 | 否 |
| name | String(255) | 教材名称 | 否 |
| awad\_info | String(255) | 获奖信息 | 是 |

#### 关联模型

* authors: 定义了与Users模型的多对多关系，通过BookAuthorAssociation表进行关联，表示教材的作者。

#### 构造函数说明

* \_\_init\_\_(self, publication\_year, name, awad\_info): 构造函数用于初始化一个新的TextBooks实例，需要提供教材的出版年份、名称和获奖信息。

#### 多对多关系

* 教材与作者之间存在多对多关系，即一本教材可以有多个作者，一个作者也可以编写多本教材。

#### 默认值

* awad\_info字段可以为空，用于记录教材的获奖情况，如果没有获奖则不填写。

#### 索引建议

* publication\_year和name: 可以根据查询需求考虑设置为索引，以优化基于出版年份或教材名称的查询效率。

### TeachingReformTeacherAssociation 模型

#### 描述

TeachingReformTeacherAssociation 模型用于建立教学改革项目与教师之间的关联关系，实现多对多的关联方式。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 关联记录 ID，主键 | 否 |
| user\_id | String(20) | 用户工号，外键 | 否 |
| project\_id | Integer | 教学改革项目 ID，外键 | 否 |

#### 外键关系

* user\_id: 与Users表的work\_id字段相关联，标识参与教学改革项目的教师。
* project\_id: 与TeachingReformProjects表的id字段相关联，标识具体的教学改革项目。

#### 模型功能

* 该模型允许将教师与其参与的教学改革项目进行关联，支持一个教师关联多个教学改革项目，以及一个教学改革项目被多个教师共同参与。

#### 约束条件

* user\_id和project\_id字段由于是外键，因此不能为空，确保每条记录都有明确的教师和教学改革项目的关联。

### TeachingReformProjects 模型

#### 描述

TeachingReformProjects 模型用于存储教学改革项目的详细信息，包括批准年份、项目名称、级别和项目描述。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 项目 ID，主键 | 否 |
| approval\_year | String(50) | 批准年份 | 否 |
| name | String(255) | 项目名称 | 否 |
| level | String(50) | 项目级别 | 否 |
| description | Text | 项目描述 | 是 |

#### 关联模型

* owner: 定义了与Users模型的多对多关系，通过TeachingReformTeacherAssociation表进行关联，表示参与教学改革项目的教师。

#### 构造函数说明

* \_\_init\_\_(self, approval\_year, level, name, description): 构造函数用于初始化一个新的TeachingReformProjects实例，需要提供项目的批准年份、级别、名称和描述。

#### 多对多关系

* 教学改革项目与教师之间存在多对多关系，即一个项目可以有多个教师参与，一个教师也可以参与多个项目。

#### 默认值

* description字段可以为空，如果没有额外的描述信息。

#### 索引建议

* approval\_year和name: 可以根据查询需求考虑设置为索引，以优化基于批准年份或项目名称的查询效率。

### ResearchAchievementAuthorAssociation 模型

#### 描述

ResearchAchievementAuthorAssociation 模型用于建立科研成果与作者（教师）之间的关联关系，实现一对多的关联方式。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 关联记录 ID，主键 | 否 |
| user\_id | String(20) | 用户工号，外键 | 否 |
| project\_id | Integer | 科研成果 ID，外键 | 否 |

#### 外键关系

* user\_id: 与 Users 表的 work\_id 字段相关联，标识科研成果的作者。
* project\_id: 与 ResearchAchievements 表的 id 字段相关联，标识具体的科研成果。

#### 模型功能

* 该模型允许将科研成果与其作者进行关联，支持一个科研成果有多个作者，以及一个作者可以有多个科研成果。

#### 约束条件

* user\_id 和 project\_id 字段由于是外键，因此不能为空，确保每条记录都有明确的作者和科研成果的关联。

### ResearchAchievements 模型

#### 描述

ResearchAchievements 模型用于存储科研成果的详细信息，包括成果年份、名称、级别和描述。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 科研成果 ID，主键 | 否 |
| achievement\_year | String(50) | 成果年份 | 否 |
| name | String(255) | 成果名称 | 否 |
| level | String(50) | 级别 | 否 |
| description | Text | 成果描述 | 是 |

#### 关联模型

* authors: 定义了与Users模型的多对多关系，通过ResearchAchievementAuthorAssociation表进行关联，表示科研成果的作者。

#### 构造函数说明

* \_\_init\_\_(self, achievement\_year, level, name, description): 构造函数用于初始化一个新的ResearchAchievements实例，需要提供成果的年份、级别、名称和描述。

#### 多对多关系

* 科研成果与作者之间存在多对多关系，即一个成果可以有多个作者，一个作者也可以有多个科研成果。

#### 默认值

* description字段可以为空，如果没有额外的描述信息。

#### 索引建议

* achievement\_year和name: 可以根据查询需求考虑设置为索引，以优化基于成果年份或名称的查询效率。

### PatentInventorAssociation 模型

#### 描述

PatentInventorAssociation 模型用于建立专利与其发明者（教师或学生）之间的关联关系，实现多对多的关联方式。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 关联记录 ID，主键 | 否 |
| project\_id | Integer | 专利 ID，外键 | 否 |
| user\_id | String(20) | 用户工号，外键 | 否 |

#### 外键关系

* project\_id: 与Patents表的id字段相关联，标识具体的专利。
* user\_id: 与Users表的work\_id字段相关联，标识专利的发明者。

#### 模型功能

* 该模型允许将专利与其发明者进行关联，支持一项专利有多个发明者，以及一个发明者可以拥有多项专利。

#### 约束条件

* project\_id和user\_id字段由于是外键，因此不能为空，确保每条记录都有明确的专利和发明者的关联。

### Patents 模型

#### 描述

Patents 模型用于存储专利的详细信息，包括申请年份、专利名称、申请号、发明人姓名集合、专利权人姓名集合以及专利状态。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 | 唯一性 | 索引 |
| --- | --- | --- | --- | --- | --- |
| id | Integer | 专利 ID，主键 | 否 |  |  |
| application\_year | Date | 申请年份 | 否 |  |  |
| title | String(255) | 专利名称 | 否 |  |  |
| application\_number | String(100) | 专利申请号 | 否 | 是 | 是 |
| status | String(50) | 专利状态（如：申请、授权等） | 否 |  |  |
| inventors\_names | String(255) | 专利发明人姓名集合 | 是 |  |  |
| shareholders | String(255) | 专利权人姓名集合 | 是 |  |  |

#### 关联模型

* inventors: 定义了与Users模型的多对多关系，通过PatentInventorAssociation表进行关联，表示专利的发明者。

#### 构造函数说明

* \_\_init\_\_(self, application\_year, title, application\_number, inventors\_names, shareholders, status): 构造函数用于初始化一个新的Patents实例，需要提供申请年份、专利名称、申请号、发明人姓名集合、专利权人姓名集合以及专利状态。

#### 多对多关系

* 专利与发明人之间存在多对多关系，即一项专利可以有多个发明人，一个发明人也可以拥有多项专利。

#### 默认值

* inventors\_names 和 shareholders 字段可以为空，用于记录专利的发明人和专利权人的姓名集合。

#### 索引建议

* application\_number: 由于是专利申请号，建议设置为唯一索引，以确保每项专利的申请号在数据库中唯一。

### CopyrightAuthorAssociation 模型

#### 描述

CopyrightAuthorAssociation 模型用于建立版权作品与其作者之间的关联关系，实现一对一或多对一的关联方式。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 关联记录 ID，主键 | 否 |
| project\_id | Integer | 版权作品 ID，外键 | 否 |
| user\_id | String(20) | 用户工号，外键 | 否 |

#### 外键关系

* project\_id: 与Copyrights表的id字段相关联，标识具体的版权作品。
* user\_id: 与Users表的work\_id字段相关联，标识版权作品的作者。

#### 模型功能

* 该模型允许将版权作品与其作者进行关联，支持一个版权作品对应一个或多个作者的情况。

#### 约束条件

* project\_id和user\_id字段由于是外键，因此不能为空，确保每条记录都有明确的版权作品和作者的关联。

### Copyrights 模型

#### 描述

Copyrights 模型用于存储软件著作权的详细信息，包括注册号、登记号、软件名称以及著作权人信息。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 著作权 ID，主键 | 否 |
| registration\_id | String(100) | 注册号 | 否 |
| registration\_number | String(100) | 登记号 | 否 |
| software\_name | String(255) | 软件名称 | 否 |
| authors\_names | String(255) | 著作权人姓名集合 | 是 |

#### 关联模型

* authors: 定义了与Users模型的多对多关系，通过CopyrightAuthorAssociation表进行关联，表示软件的著作权人。

#### 构造函数说明

* \_\_init\_\_(self, registration\_id, registration\_number, software\_name, authors\_names): 构造函数用于初始化一个新的Copyrights实例，需要提供注册号、登记号、软件名称以及著作权人姓名集合。

#### 多对多关系

* 软件著作权与著作权人之间存在多对多关系，即一个软件可以有多个著作权人，一个著作权人也可以拥有多个软件著作权。

#### 默认值

* authors\_names 字段可以为空，用于记录软件的著作权人姓名集合。

#### 索引建议

* registration\_id 和 registration\_number: 可以根据查询需求考虑设置为索引，以优化基于注册号或登记号的查询效率。

### AdmissionInfo 模型

#### 描述

AdmissionInfo 模型用于存储招生信息的详细描述，包括招生类别、技术要求、学习形式、工作时间、其他要求、联系人和联系信息。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 招生信息 ID，主键 | 否 |
| category | String(50) | 招生类别 | 否 |
| technical\_requirements | Text | 技术要求 | 是 |
| study\_mode | String(50) | 学习形式 | 是 |
| work\_schedule | String(100) | 工作时间 | 是 |
| other\_requirements | Text | 其他要求 | 是 |
| contact\_person | String(50) | 联系人 | 是 |
| contact\_information | String(100) | 联系信息 | 是 |

#### 模型功能

* AdmissionInfo模型允许教育机构或招生部门记录和存储有关招生过程的各种信息，以供潜在申请者查询。

#### 默认值

* 所有字段均没有默认值，根据招生信息的具体内容进行填写。

#### 索引建议

* category: 如果该字段经常用于查询，可以考虑设置索引以提高查询效率。

### InternationalPartnership 模型

#### 描述

InternationalPartnership 模型用于存储国际合作伙伴关系的信息，包括合作伙伴的名称、所属国家、合作项目、合作时间、状态和项目描述。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 合作伙伴关系 ID，主键 | 否 |
| name | String(255) | 大学/企业名称 | 否 |
| country | String(100) | 所属国家 | 否 |
| project | String(255) | 合作项目 | 否 |
| start\_date | Date | 合作开始时间 | 是 |
| end\_date | Date | 合作结束时间 | 是 |
| status | String(50) | 合作状态（如：进行中、已完成等） | 是 |
| description | Text | 合作项目描述或详情 | 是 |

#### 模型功能

* InternationalPartnership模型允许存储和管理国际合作伙伴的详细信息，便于跟踪和回顾合作的历程和状态。

#### 默认值

* 所有日期字段start\_date和end\_date以及状态status和描述description字段都可以为空，允许在记录合作时不立即指定这些信息。

#### 索引建议

* name和country: 可以根据查询需求考虑设置为索引，以优化基于合作伙伴名称或国家的查询效率。

### Books 模型

#### 描述

Books 模型用于存储图书馆或图书管理系统中的图书信息，包括图书 ID、名称、作者、出版年份、位置和可借状态。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 | 唯一性 | 索引 |
| --- | --- | --- | --- | --- | --- |
| id | Integer | 唯一标识符，主键 | 否 |  |  |
| book\_id | String(100) | 图书 ID，唯一值 | 否 | 是 | 是 |
| name | String(100) | 图书名称 | 否 |  |  |
| authors | String(100) | 作者 | 是 |  |  |
| publish\_year | String(50) | 出版年份 | 是 |  |  |
| location | String(50) | 图书当前位置 | 是 |  |  |
| available | Boolean | 图书是否可借 | 否 |  |  |

#### 模型功能

* Books模型提供了图书的基本属性，适用于图书馆或图书管理系统中对图书的登记和管理。

#### 默认值

* available 字段默认为 True，表示图书默认情况下是可借的。

#### 索引建议

* book\_id: 由于是图书的唯一标识，建议设置为唯一索引，以优化查找效率。

#### 序列化方法

* to\_json(self): 方法将图书对象序列化为 JSON 格式，包括图书的所有核心属性。

序列化示例：

{  
 'id': '图书的唯一ID',  
 'name': '图书名称',  
 'authors': '作者',  
 'publish\_year': '出版年份',  
 'location': '图书当前位置',  
 'available': true 或 false  
}

### BookLoans 模型

#### 描述

BookLoans 模型用于记录图书借阅的详细信息，包括借阅的图书、用户、借阅和应还日期以及借阅状态。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 借阅记录 ID，主键 | 否 |
| book\_id | String(100) | 借阅图书 ID，外键关联Books表 | 否 |
| user\_id | String(20) | 借阅用户 ID，外键关联Users表 | 否 |
| brows\_request\_id | Integer | 预借请求 ID，外键关联BookLoanRequest表 | 是 |
| return\_request\_id | Integer | 还书请求 ID，外键关联BookLoanRequest表 | 是 |
| should\_return\_date | DateTime | 应还日期 | 否 |
| loan\_date | DateTime | 借阅日期 | 否 |
| return\_date | DateTime | 实际还书日期 | 是 |
| status | String(20) | 借阅状态 | 否 |

#### 关联模型

* book: 与Books模型的book\_id字段关联，表示被借阅的图书。
* requester: 与Users模型的user\_id字段关联，表示借阅图书的用户。
* brows\_request: 与BookLoanRequest模型的brows\_request\_id字段关联，表示预借请求。
* return\_request: 与BookLoanRequest模型的return\_request\_id字段关联，表示还书请求。

#### 模型方法

* get\_left\_days(self): 计算并返回剩余还书天数。
* to\_json(self): 将借阅记录对象序列化为 JSON 格式。

序列化示例：

{  
 "id": "借阅记录ID",  
 "book\_id": "借阅图书ID",  
 "user\_id": "用户ID",  
 "loan\_date": "借阅日期",  
 "return\_date": "实际还书日期",  
 "status": "借阅状态"  
}

### ViolationRecords 模型

#### 描述

ViolationRecords 模型用于记录用户在图书借阅过程中的违规行为，包括违规日期、描述以及与用户和借阅记录的关联。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 违规记录 ID，主键 | 否 |
| user\_id | String(20) | 违规用户 ID，外键关联Users表 | 否 |
| loan\_id | Integer | 借阅记录 ID，外键关联BookLoans表 | 否 |
| violation\_date | DateTime | 违规日期 | 否 |
| description | Text | 违规行为描述 | 否 |

#### 关联模型

* loans: 与BookLoans模型的loan\_id字段关联，表示对应的借阅记录。
* user: 与Users模型的user\_id字段关联，表示发生违规行为的用户。

#### 外键关系

* user\_id: 与Users表的work\_id字段相关联，标识违规的用户。
* loan\_id: 与BookLoans表的id字段相关联，标识具体的借阅记录。

#### 模型功能

* ViolationRecords模型允许图书馆或图书管理系统记录用户的违规行为，并与用户及借阅记录进行关联，便于追踪和管理。

#### 默认值

* 无特定字段默认值，所有字段根据违规事件的具体信息进行填写。

#### 索引建议

* user\_id和loan\_id: 可以考虑设置索引，以提高基于用户或借阅记录的查询效率。

#### 示例

违规记录的创建通常涉及提供用户 ID、借阅记录 ID、违规日期以及违规行为的详细描述。例如：

用户ID: U1234  
借阅记录ID: L5678  
违规日期: 2024-06-07T14:23:00  
违规行为描述: 超过应还日期未归还图书  
### 图书借阅请求表

#### 描述

图书借阅请求表用于存储用户对图书的借阅或归还请求信息，包括请求人 ID、处理人 ID、图书 ID、请求日期、处理日期、状态、申请理由和处理备注。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 | 默认值 | 关联模型 |
| --- | --- | --- | --- | --- | --- |
| **id** | Integer | 请求记录 ID，主键 | 否 | 自增 |  |
| **requester\_id** | Integer | 请求人 ID，外键 | 否 | None | 用户表 |
| **processor\_id** | Integer | 处理人 ID，外键 | 是 | None | 用户表 |
| **book\_id** | Integer | 图书 ID，外键 | 否 | None | 图书信息表 |
| **request\_date** | DateTime | 请求日期 | 否 | 当前时间戳 |  |
| **process\_date** | DateTime | 处理日期 | 是 | None |  |
| **status** | String(50) | 请求状态 | 否 | ‘待处理’ |  |
| **request\_reason** | Text | 申请理由 | 是 | None |  |
| **processing\_note** | Text | 处理备注 | 是 | None |  |
| **request\_type** | String(10) | 申请类型，如 借阅、还书 | 否 | None |  |

#### 反向引用

* requester: 通过用户表关联请求人信息，允许从请求记录访问请求人详情。
* processor: 通过用户表关联处理人信息，允许从请求记录访问处理人详情。
* book: 通过图书信息表关联图书信息，允许从请求记录访问图书详情。

#### 初始化方法

* \_\_init\_\_(self, requester\_id, book\_id, request\_type, request\_reason, request\_date, status='待处理', processing\_note=None, processor\_id=None, process\_date=None): 初始化借阅请求记录。

#### 方法说明

* to\_dict(self): 将借阅请求记录转换为字典格式，方便进行数据交换和展示。
* \_\_repr\_\_(self): 提供对象的字符串表示，方便调试和日志记录。

#### 默认值

* request\_date: 默认为当前时间戳。

#### 注意事项

* requester\_id 字段作为外键关联用户表，用于标识发起请求的用户。
* processor\_id 字段作为外键关联用户表，用于标识处理请求的用户，可以为空，表示可能尚未分配处理人或处理已完成。
* status 字段用于标识请求的当前状态，如“待处理”、“已批准”、“已拒绝”等。
* request\_type 字段用于标识请求的类型，如借阅或还书。

### BookLoanRequest 模型

#### 描述

BookLoanRequest 模型用于管理图书借阅请求的详细信息，包括请求者、处理人、图书、请求日期、处理日期、状态、申请理由、处理备注和请求类型。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 | 默认值 |
| --- | --- | --- | --- | --- |
| id | Integer | 请求记录 ID，主键 | 否 |  |
| requester\_id | String(20) | 请求者 ID，外键关联Users表 | 否 |  |
| processor\_id | String(20) | 处理人 ID，外键关联Users表 | 是 |  |
| book\_id | String(100) | 图书 ID，外键关联Books表 | 否 |  |
| request\_date | DateTime | 请求日期 | 否 | 当前时间 (datetime.now) |
| process\_date | DateTime | 处理日期 | 是 |  |
| status | String(50) | 请求状态 | 否 | '待处理' |
| request\_reason | Text | 申请理由 |  |  |
| processing\_note | Text | 处理备注 |  |  |
| request\_type | String(10) | 申请类型，如 借阅、还书 | 否 |  |

#### 关联模型

* requester: 与Users模型的requester\_id字段关联，表示提交请求的用户。
* processor: 与Users模型的processor\_id字段关联，表示处理请求的用户。
* book: 与Books模型的book\_id字段关联，表示请求借阅或归还的图书。

#### 构造函数说明

* \_\_init\_\_(self, requester\_id, book\_id, request\_type, request\_reason, request\_date, status='待处理', processing\_note=None, processor\_id=None, process\_date=None): 构造函数用于初始化一个新的BookLoanRequest实例，需要提供请求者 ID、图书 ID、请求类型、申请理由、请求日期，其他字段可选。

#### 序列化方法

* to\_dict(self): 方法将图书借阅请求对象序列化为字典格式，包括请求的所有核心属性和关联对象的信息。

序列化示例：

{  
 "id": "请求记录ID",  
 "requester\_id": "请求者ID",  
 "processor\_id": "处理人ID",  
 "book\_id": "图书ID",  
 "request\_date": "请求日期",  
 "process\_date": "处理日期",  
 "status": "请求状态",  
 "request\_reason": "申请理由",  
 "processing\_note": "处理备注",  
 "request\_type": "申请类型",  
 "requester": "请求者信息字典",  
 "requester\_name": "请求者用户名",  
 "book": "图书信息JSON"  
}

### LibraryStatus 模型

#### 描述

LibraryStatus 模型用于定义图书馆管理的常量设置，包括间隔日期、借阅期限、超出期限提醒天数、单个用户借阅数量限制和单个用户违规记录数量限制等。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 默认值 | 是否可为空 |
| --- | --- | --- | --- | --- |
| id | Integer | 常量设置 ID，主键 |  | 否 |
| interval\_date | Integer | 间隔日期（天数） | 5 | 否 |
| borrow\_period | Integer | 借阅期限（天数） | 30 | 否 |
| overdue\_reminder\_days | Integer | 超出期限提醒天数 | 3 | 否 |
| borrow\_limit | Integer | 单个用户借阅数量限制 | 2 | 否 |
| violation\_limit | Integer | 单个用户违规记录数量限制 | 3 | 否 |
| is\_book\_admin | Boolean | 是否图书管理员 | False | 否 |

#### 序列化方法

* to\_dict(self): 方法将图书馆常量设置对象序列化为字典格式，包含所有常量设置的当前值。

序列化示例：

{  
 "id": "常量设置ID",  
 "interval\_date": "间隔日期天数",  
 "borrow\_period": "借阅期限天数",  
 "overdue\_reminder\_days": "超出期限提醒天数",  
 "borrow\_limit": "单个用户借阅数量限制",  
 "violation\_limit": "单个用户违规记录数量限制",  
 "is\_book\_admin": "是否图书管理员"  
}

### ResourceDownload 模型

#### 描述

ResourceDownload 模型用于存储可供下载的资源信息，包括资源名称、下载链接、类型、作者以及简介。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 |
| --- | --- | --- | --- |
| id | Integer | 资源 ID，主键 | 否 |
| name | Text | 资源名称 | 否 |
| url | String(200) | 资源下载链接 | 是 |
| type | String(20) | 资料类型 | 否 |
| work\_id | String(20) | 上传者工号，外键关联Users表 | 否 |
| author | String(100) | 作者 | 是 |
| introduction | Text | 简介 | 是 |

#### 序列化方法

* to\_dict(self): 方法将资源下载对象序列化为字典格式，包含资源的基本信息。

序列化示例：

{  
 "id": "资源ID",  
 "name": "资源名称",  
 "url": "资源下载链接",  
 "type": "资料类型",  
 "work\_id": "上传者工号",  
 "author": "作者信息"  
}

### News 模型

#### 描述

News 模型用于存储新闻或通知的详细信息，包括标题、内容、链接、类别、作者、创建时间、发布时间、附件和封面。

#### 字段说明

| 字段名 | 数据类型 | 描述 | 是否可为空 | 默认值 |
| --- | --- | --- | --- | --- |
| id | Integer | 新闻 ID，主键 | 否 |  |
| title | String(255) | 标题 | 否 |  |
| content | Text | 内容 | 是 |  |
| link | String(255) | 外部链接 | 是 |  |
| category | String(100) | 类别 | 否 |  |
| author | String(100) | 作者 | 否 |  |
| create\_time | DateTime | 创建时间 | 否 | 当前时间 (datetime.now) |
| publish\_time | DateTime | 发布时间 | 是 |  |
| attachments | ARRAY(String) | 附件地址数组 | 是 |  |
| cover | String(255) | 封面属性 | 否 |  |

#### 序列化方法

* to\_dict(self): 方法将新闻对象序列化为字典格式，包含新闻的所有核心属性。

序列化示例：

{  
 "id": "新闻ID",  
 "title": "新闻标题",  
 "content": "新闻内容",  
 "link": "外部链接",  
 "category": "新闻类别",  
 "author": "作者",  
 "create\_time": "创建时间",  
 "publish\_time": "发布时间",  
 "attachments": ["附件地址1", "附件地址2"],  
 "cover": "封面属性"  
}

## api 文档

### Blueprint: loans\_bp

#### 路径: /api/books\_loans/request\_borrow/

##### 方法: POST

##### 功能: 申请借阅图书

###### 参数

* book\_id (string): 图书 ID

###### 响应

* 201: 借阅申请已提交，等待审批
* 400: 您已经达到最大借阅数量 或 您已提交过借阅申请，请等待审批
* 404: 图书不存在或不可申请
* 500: 借阅申请失败

###### 请求体

* reason (string, 可选): 借阅原因

#### 路径: /api/books\_loans/request\_return/

##### 方法: POST

##### 功能: 申请归还图书

###### 参数

* loan\_id (int): 借阅记录 ID

###### 响应

* 201: 还书申请已提交，等待审批
* 400: 您已提交过还书申请，请等待审批
* 403: 您没有这本书的借阅记录
* 404: 借阅记录不存在
* 500: 还书申请失败

###### 请求体

* reason (string, 可选): 归还原因

#### 路径: /api/books\_loans/mybooks

##### 方法: GET

##### 功能: 获取当前用户所借阅的所有书籍

##### 响应

* 200: 返回用户所借阅的书籍列表

#### 路径: /api/books\_loans/mycredit

##### 方法: GET

##### 功能: 获取当前用户的诚信记录

##### 响应

* 200: 返回用户的诚信记录列表

### Blueprint: books\_bp

#### 路径: /api/books

##### 方法: GET

##### 功能: 获取所有图书的信息

##### 响应

* 200: 返回所有图书的列表

#### 路径: /api/books/

##### 方法: GET

##### 功能: 获取特定图书的信息

##### 参数

* book\_id (int): 图书的 ID

##### 响应

* 200: 返回特定图书的信息
* 404: 如果图书不存在，返回错误信息

#### 路径: /api/books/

##### 方法: PATCH

##### 功能: 更新图书的信息

##### 参数

* book\_id (int): 图书的 ID

##### 请求体

* barcode (string, 可选): 图书的条形码
* name (string, 可选): 图书的名称
* authors (string, 可选): 图书的作者
* publish\_year (int, 可选): 图书的出版年份
* location (string, 可选): 图书的位置
* available (boolean, 可选): 图书是否可用

##### 响应

* 200: 更新图书信息成功
* 400: 如果请求体中缺少必要的参数，返回错误信息
* 404: 如果图书不存在，返回错误信息

#### 路径: /api/books

##### 方法: POST

##### 功能: 创建新的图书

##### 请求体

* barcode (string): 图书的条形码
* name (string): 图书的名称
* authors (string): 图书的作者
* publish\_year (int): 图书的出版年份
* location (string, 可选): 图书的位置
* available (boolean, 可选): 图书是否可用

##### 响应

* 201: 创建图书成功
* 400: 如果请求体中缺少必要的参数，返回错误信息

#### 路径: /api/books/

##### 方法: PUT

##### 功能: 完全更新图书的信息

##### 参数

* book\_id (int): 图书的 ID

##### 请求体

* barcode (string, 可选): 图书的条形码
* name (string, 可选): 图书的名称
* authors (string, 可选): 图书的作者
* publish\_year (int, 可选): 图书的出版年份
* location (string, 可选): 图书的位置
* available (boolean, 可选): 图书是否可用

##### 响应

* 200: 更新图书信息成功
* 400: 如果请求体中缺少必要的参数，返回错误信息
* 404: 如果图书不存在，返回错误信息

#### 路径: /api/books/

##### 方法: DELETE

##### 功能: 删除图书

##### 参数

* book\_id (int): 图书的 ID

##### 响应

* 204: 删除图书成功
* 404: 如果图书不存在，返回错误信息

### Blueprint: introduction

#### 路径: /api/introduction\_update

##### 方法: GET

##### 功能: 获取所有简介

##### 响应

* 200: 返回所有简介的列表

#### 路径: /api/introduction\_update/

##### 方法: POST

##### 功能: 建立新的简介

##### 请求体

* name (string): 简介的名称
* short\_name (string): 简介的简称
* introduction (string): 简介的介绍内容
* address (string): 地址
* phone (string): 电话
* fax (string): 传真
* email (string): 电子邮件
* website (string): 网站
* logo\_path (string): logo 路径
* is\_active (boolean): 是否激活
* operator\_id (int): 操作员 ID

##### 响应

* 200: 新建简介成功

#### 路径: /api/introduction\_update/

##### 方法: POST

##### 功能: 切换简介的激活状态

##### 参数

* id (int): 简介的 ID

##### 请求体

* is\_active (boolean): 简介的激活状态

##### 响应

* 200: 简介状态更新成功
* 404: 如果简介不存在，返回错误信息

#### 路径: /api/introduction\_update/active

##### 方法: GET

##### 功能: 获取激活的简介

##### 响应

* 200: 返回激活的简介信息
* 404: 如果没有激活的简介，返回错误信息

### Blueprint: libraryS\_setting

#### 路径: /api/libraryS\_setting/get\_libraryS\_status

##### 方法: GET

##### 功能: 获取图书馆状态信息

##### 参数

* page (int): 页码
* per\_page (int): 每页显示数量

##### 响应

* 200: 返回图书馆状态信息的分页列表

#### 路径: /api/libraryS\_setting/modify\_libraryS\_status

##### 方法: POST

##### 功能: 修改图书馆状态

##### 请求体

* work\_id (string): 用户工作 ID
* interval\_date (int, 可选): 借阅间隔日期
* borrow\_period (int, 可选): 借阅期限
* overdue\_reminder\_days (int, 可选): 逾期提醒天数
* borrow\_limit (int, 可选): 借阅限制
* violation\_limit (int, 可选): 违规限制
* is\_book\_admin (boolean, 可选): 是否为图书管理员

##### 响应

* 200: 修改成功
* 401: 用户不存在
* 500: 修改失败

### Blueprint: materials\_bp

#### 路径: /api/materials

##### 方法: POST

##### 功能: 上传文件

##### 请求体

* name (string): 文件名称
* type (string): 文件类型
* work\_id (string): 工作 ID
* author (string): 作者
* introduction (string): 简介
* file (file): 要上传的文件

##### 响应

* 201: 文件上传成功
* 400: 没有文件部分 或 没有选择文件 或 不支持的文件类型
* 415: 不支持的文件类型
* 500: 文件上传失败

#### 路径: /api/materials/files

##### 方法: GET

##### 功能: 根据类型获取文件列表

##### 参数

* type (string): 文件类型
* page (int): 页码
* limit (int): 每页显示数量

##### 响应

* 200: 返回文件列表和分页信息
* 400: 缺少文件类型参数

### Blueprint: news\_bp

#### 路径: /api/news

##### 方法: POST

##### 功能: 上传图片

##### 请求体

* image (file): 要上传的图片文件

##### 响应

* 200: 图片上传成功，返回图片的 URL 路径
* 400: 没有文件部分 或 没有选择文件 或 不支持的文件类型

#### 路径: /api/news

##### 方法: POST

##### 功能: 提交新闻

##### 请求体

* title (string): 新闻标题
* author (string): 新闻作者
* cover (file): 新闻封面图片
* content (string): 新闻内容
* category (string): 新闻分类
* attachmentLink (string): 附件链接（多个用逗号分隔）
* files (file): 附件文件（多个）

##### 响应

* 200: 新闻内容提交成功

#### 路径: /api/news/news-list

##### 方法: GET

##### 功能: 获取新闻列表

##### 响应

* 200: 返回最新的 10 条新闻列表

#### 路径: /api/news/

##### 方法: GET

##### 功能: 获取单条新闻内容

##### 参数

* news\_id (int): 新闻 ID

##### 响应

* 200: 返回单条新闻内容
* 404: 如果新闻不存在，返回错误信息

#### 路径: /api/news/category-news

##### 方法: GET

##### 功能: 获取分类新闻

##### 响应

* 200: 返回指定分类的最新一条新闻

### Blueprint: request\_api

#### 路径: /api/request\_process

##### 方法: PUT

##### 功能: 处理借阅请求

##### 参数

* request\_id (int): 借阅请求 ID

##### 请求体

* action (string): 操作类型（‘同意’ 或 ‘拒绝’）
* note (string, 可选): 处理备注

##### 响应

* 200: 图书借阅请求已处理
* 403: 您没有权限处理此请求
* 404: 请求不存在或已处理
* 400: 无效的操作
* 500: 处理请求失败

#### 路径: /api/request\_process/return

##### 方法: PUT

##### 功能: 处理归还请求

##### 参数

* request\_id (int): 归还请求 ID

##### 请求体

* action (string): 操作类型（‘同意’ 或 ‘拒绝’）
* note (string, 可选): 处理备注

##### 响应

* 200: 图书归还请求已处理
* 403: 您没有权限处理此请求
* 404: 请求不存在或已处理
* 400: 无效的操作
* 500: 处理请求失败

#### 路径: /api/request\_process/all

##### 方法: GET

##### 功能: 获取所有待处理的请求

##### 响应

* 200: 返回所有待处理的请求列表

#### 路径: /api/request\_process/my

##### 方法: GET

##### 功能: 获取所有由自己发起的请求

##### 响应

* 200: 返回所有由自己发起的请求列表

### Blueprint: users\_bp

#### 路径: /api/users

##### 方法: PUT

##### 功能: 更新用户信息

##### 请求体

* field (string): 要更新的字段
* value (string): 更新的值

##### 响应

* 200: 用户信息更新成功
* 400: 缺少必要的字段
* 404: 用户不存在
* 500: 更新失败

#### 路径: /api/users/avatar

##### 方法: POST

##### 功能: 更新用户头像

##### 请求体

* image (file): 要上传的头像文件

##### 响应

* 200: 头像更新成功，返回头像的 URL 路径
* 400: 没有文件部分 或 没有选择文件 或 不支持的文件类型
* 404: 用户不存在
* 500: 更新失败

#### 路径: /api/users/avatar

##### 方法: GET

##### 功能: 获取用户头像地址

##### 响应

* 200: 返回用户头像的 URL 路径
* 404: 如果用户不存在，返回错误信息

### 错误处理器

#### 404 Not Found

返回一个 JSON 对象，包含错误信息。

#### 400 Bad Request

返回一个 JSON 对象，包含错误信息和描述。

## 待添加

* 动态消息表：（News）
* 文件下载 blacklist 表：（FileDownloadBlacklist）

## 视图

### 普通视图

* 视图的本质是虚拟表，通过 SQL 语句来查询数据，而不是直接查询数据库中的表， 所以还是翻译成 SQL 语句来与数据库进行交互
* 本项目使用的是 SQLAlchemy 来进行 ORM 映射， 也是翻译成 SQL 语句来与数据库进行交互
* 所以本项目没有创建视图的需要

### 物化视图

* 物化视图是一种特殊的视图，它将数据保存在物理表中，而不是在每次查询时都重新计算，从而提高查询效率。
* 物化视图的创建需要手动执行，在创建视图的同时，系统会自动将视图数据保存到物理表中。
* 本项目数据量较小，不需要创建物化视图

### 综上所述，本项目没有创建视图的需要

## 索引

* 本项目使用的是 SQLAlchemy 来进行 ORM 映射，表所对应的实体类中，有外键关联的实体类，会自动生成索引。
* 所以本项目没有创建索引的需要