数据库设计说明书

一、引言

1.1. 编写目的

本文档旨在明确智汇选课的数据库系统的详细设计，包括数据库的逻辑结构、物理结构、表结构、关系、约束、安全及性能考虑等。它为后续的数据库创建、维护以及应用程序开发提供技术依据，并确保开发团队对数据库结构有统一、准确的理解。

预期读者： 数据库管理员（DBA）、后端开发工程师、测试工程师及项目管理人员。

1.2. 项目背景

当前高校选课环节存在信息不对称问题，影响学生选课决策与学习体验。学生获取课程信息的渠道单一，多依赖官方发布的简略大纲，缺乏真实学习体验参考，对课程难度、考核方式、教学风格等关键内容认知模糊，易因预期与实际偏差导致选课盲目，进而影响学习积极性与成绩表现。

同时，学生间的课程经验分享多分散在社交平台、班级群等非官方渠道，信息碎片化且真实性难以验证，存在误导风险。此外，教师难以系统性收集学生对课程的反馈意见，无法及时掌握教学过程中的问题与不足，导致教学优化缺乏精准的数据支撑，形成“学生选课难、教师改进难”的双重困境。

1.3. 定义、缩略语和术语

- DDL： 数据定义语言，用于定义和管理数据库对象。
- DML： 数据操作语言，用于查询和操纵数据。
- 主键（Primary Key）： 唯一标识表中每条记录的字段或字段组合。
- 外键（Foreign Key）： 用于建立和加强两个表数据之间链接的一列或多列。
- 索引（Index）： 用于加速数据检索的数据库对象。
- 视图 (View)：是一种虚拟表，其内容由查询定义。
- 存储过程（Procedure）：是一组预先编写并存储在数据库中的SQL语句集合，用于完成特定的数据库操作。
- 触发器（Trigger）： 是一种特殊的存储过程，它会在特定事件（如 *INSERT*、*UPDATE* 或 *DELETE*）发生时自动执行。

1.4 参考资料

1. 《智汇选课需求规格说明书》
2. 《智汇选课系统设计说明书》

3. 《GB/T 8567-2006 计算机软件文档编制规范》
4. 《MySQL官方文档》

二、数据库环境说明

2.1 数据库管理系统选型

本系统选用 **MySQL 8.0** 作为数据库管理系统，选型依据如下：

1. **功能适配性**：支持复杂的关系型数据模型、外键约束、枚举类型、时间戳自动更新等核心功能，能满足系统多实体关联、数据完整性校验的需求。
2. **性能表现**：针对中小规模数据量具备优秀的读写性能，支持索引优化、查询缓存等机制，可应对学生选课评价场景下的高频数据访问（如课程查询、评论提交、资源下载等）。
3. **兼容性与生态**：与主流后端开发语言（Java、Python、PHP 等）及框架（Spring Boot、Django 等）兼容性良好，降低开发对接成本；同时具备完善的社区支持和文档资源，便于问题排查与维护。
4. **部署与成本**：开源免费，支持 Windows、Linux 等多操作系统部署，可灵活适配云端或本地服务器环境，降低项目运维成本。

2.2 运行环境要求

环境类型	具体要求
操作系统	Windows Server 2016 及以上 / CentOS 7.0 及以上 / Ubuntu 18.04 及以上
硬件配置	CPU：至少 4 核 8 线程；内存：不少于 8GB；磁盘空间：预留至少 50GB（用于数据存储、日志及备份）
网络环境	支持 TCP/IP 协议，服务器与应用服务器之间网络延迟 ≤10ms，保障数据传输稳定性
依赖组件	安装 OpenSSL（用于数据加密传输）、MySQL Connector（用于应用程序连接数据库）

2.3 数据库配置参数

为优化系统性能，建议采用以下核心配置参数（基于 MySQL 8.0）：

代码块

```
[mysqld]
# 字符集配置，支持中文等多字符存储
character-set-server = utf8mb4
collation-server = utf8mb4_unicode_ci
# 最大连接数，适配并发访问需求
max_connections = 1000
# 表缓存数量，提升表访问效率
table_open_cache = 2000
# 日志配置，便于数据追溯与问题排查
slow_query_log = ON
slow_query_log_file = /var/log/mysql/slow.log
long_query_time = 2  # 记录执行时间超过2秒的慢查询
# 数据存储引擎，InnoDB支持事务与外键
default-storage-engine = InnoDB
# 事务隔离级别，保障数据一致性
transaction_isolation = REPEATABLE-READ
```

三、数据库命名规范

3.1 通用命名原则

1. **语义化**：命名需准确反映对象含义，避免模糊或无意义的名称（如避免使用 “table1” “col1” 等）。
2. **一致性**：同一类型的对象命名风格保持统一，便于开发人员理解与维护。
3. **简洁性**：在保证语义清晰的前提下，命名尽量简洁，避免冗余。
4. **兼容性**：避免使用 MySQL 关键字（如 “order” “user” “select” 等），若必须使用需用反引号（`）包裹。
5. **大小写规范**：表名、字段名、索引名等采用小写字母，单词之间用下划线（_）分隔（蛇形命名法），避免大小写敏感导致的兼容性问题。

3.2 表命名规范

1. **命名格式**：采用 “小写英文单词 / 缩写_小写英文单词 / 缩写” 的形式，优先使用完整单词，必要时可使用通用缩写（如 “Comments” 缩写为 “comm”，但需在项目文档中统一说明）。
2. **具体规则**：
   - 实体表直接使用实体名称的复数形式（如用户表 “users”、课程表 “courses”）。
   - 关联表（多对多关系）采用 “关联实体 1_关联实体 2” 的形式，按字母顺序排列（如角色 - 权限关联表 “role_permissions”、资源 - 标签关联表 “resource_tags”）。
   - 特殊功能表根据功能命名（如信誉记录表 “reputation_records”、举报表 “reviews”）。
3. **示例**：`course_comments`（课程评论表）、`user_items`（用户 - 物品关联表）。

3.3 字段命名规范

1. **命名格式**：采用 “小写英文单词 / 缩写_小写英文单词 / 缩写” 的形式，明确字段含义。
2. **具体规则**：
   - 主键字段统一命名为 “表名缩写_id” 或直接使用 “id”（如用户表主键 “user_id”、评论表主键 “comment_id”）；关联表中引用其他表主键的外键字段，直接使用被引用表的主键字段名（如用户表主键 “user_id”，在评论表中作为外键仍命名为 “user_id”）。
   - 时间相关字段：创建时间命名为 “created_at”，更新时间命名为 “updated_at”，审核时间命名为 “reviewed_at” 等，统一使用 “_at” 后缀。
   - 状态相关字段：统一使用 “status” 命名，配合枚举类型定义具体状态值。
   - 数量相关字段：使用 “count” 作为后缀（如下载次数 “download_count”、评分人数 “rating_count”）。
3. **示例**：`password_hash`（密码哈希字段）、`average_rating`（平均评分字段）、`is_visible`（是否可见字段）。

3.4 索引命名规范

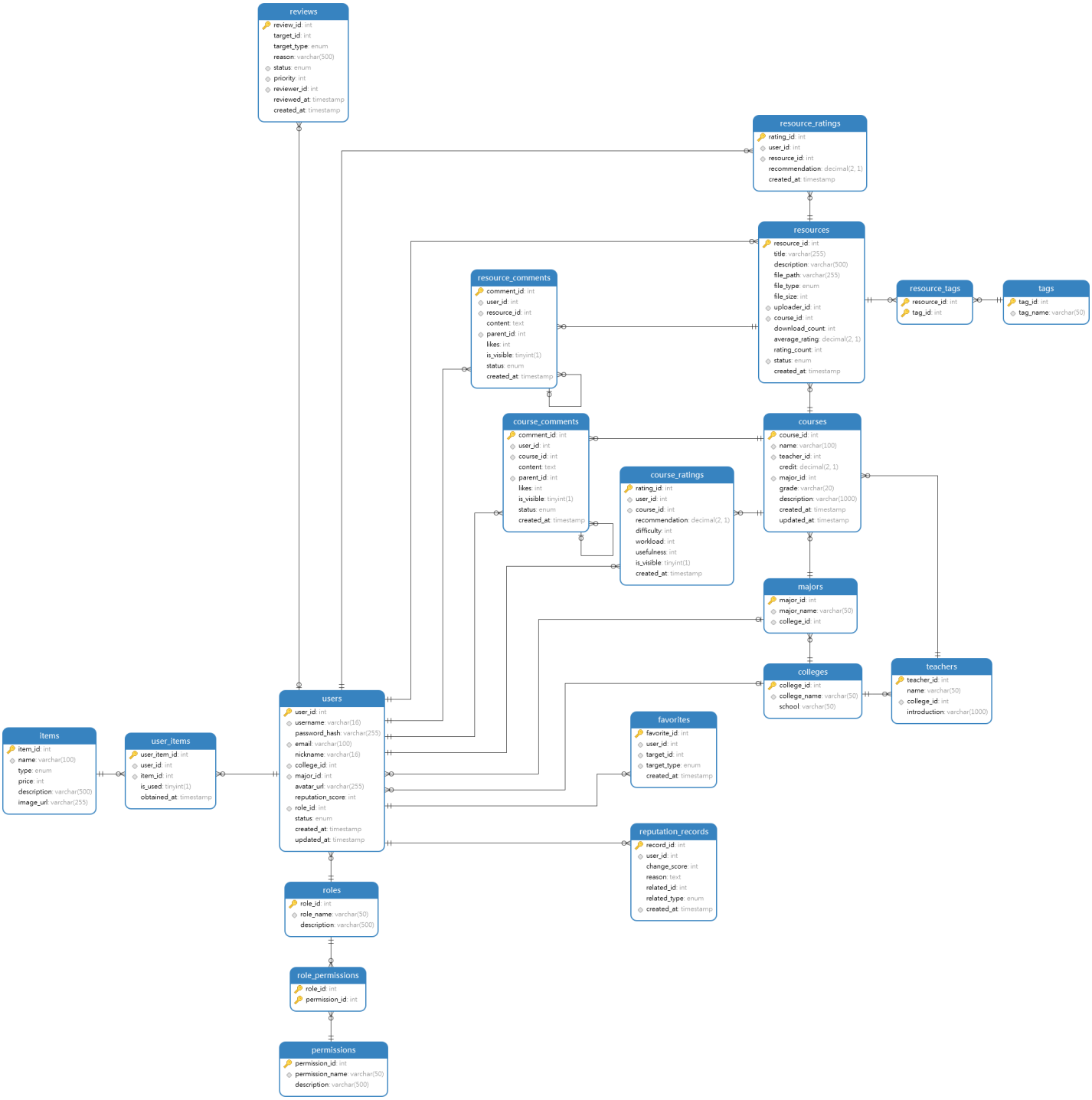
1. **命名格式**：采用 “索引类型_表名_字段名” 的形式，索引类型用缩写表示（主键索引 “pk”、唯一索引 “uk”、普通索引 “idx”、外键索引 “fk”）。
2. **具体规则**：
   - 主键索引：默认由数据库自动生成，若手动创建则命名为 “pk_表名”（如 “pk_users”）。
   - 唯一索引：命名为 “uk_表名_字段名”，多字段唯一索引用下划线连接字段名（如 “uk_users_username” “uk_favorites_user_target_type”）。
   - 普通索引：命名为 “idx_表名_字段名”，用于加速高频查询字段（如 “idx_courses_teacher_id” “idx_resources_course_id”）。
   - 外键索引：命名为 “fk_表名_字段名”（如 “fk_course_comments_user_id”）。
3. **示例**：`idx_course_ratings_course_id`（课程评分表中课程 ID 字段的普通索引）、`uk_resources_title`（资源表中标题字段的唯一索引）。

3.5 其他对象命名规范

1. **视图**：采用 “v_表名_功能描述” 的形式（如 “v_course_avg_rating”，课程平均评分视图）。
2. **存储过程**：采用 “sp_功能描述” 的形式（如 “sp_update_user_reputation”，更新用户信誉分的存储过程）。
3. **触发器**：采用 “tri_表名_触发事件” 的形式（如 “tri_users_updated_at”，用户表更新时间自动触发的触发器）。

四、逻辑结构设计

4.1. 实体关系图（ERD）



4.2. 实体清单与描述

Users（用户）：包含用户基本信息，如昵称、性别、头像、学校、学院等，是系统的核心实体，与其他多个实体存在关联。

Roles（角色）：定义用户在系统中的角色，如普通用户、管理员等，用于权限控制。

Permissions（权限）：具体的操作权限，如用户管理、课程管理等，与角色关联。

Role_Permissions（角色 - 权限关联）：多对多关系，关联角色和权限。

Courses（课程）：存储课程信息，如课程名、简介、学分等，与学院、专业等关联。

Colleges（学院）：学院信息，用于课程的分类展示。

Majors（专业）：专业信息，与学院关联，用于课程的分类。

Teachers（教师）：教师信息，负责教授课程。

Course_Comments（课程评论）：用户对教师所授课程的评论，关联用户、课程、教师。

Course_Ratings（课程评分）：用户对教师所授课程的评分，关联用户、课程、教师。

Resources（资源）：用户上传的课程资源，如课件、资料等。

Resource_Ratings（资源评分）：用户对资源的评分，关联用户、资源。

Resource_Comments（资源评论）：用户对资源的评论，关联用户、资源。

Resource_Tags（资源 - 标签关联）：多对多关系，关联资源和标签，用于资源分类。

Tags（标签）：资源的标签，方便资源搜索和分类。

Favorites（收藏）：用户收藏的课程、教师或资源，关联用户和被收藏对象。

Reviews（举报）：用户对违规资源的举报，关联用户、资源。

Reputation_Records（信誉记录）：记录用户的信誉分变化，关联用户和相关行为（如上传资源、评论被点赞等）。

Items（物品）：可兑换的虚拟物品，如头像框、勋章等。

Use_Items（用户 - 物品关联）：用户兑换的物品记录，关联用户和物品。

五、物理结构设计

5.1. 表结构设计

users表

```sql
CREATE TABLE `users` (
    `user_id` INT NOT NULL AUTO_INCREMENT COMMENT '用户唯一标识',
    `username` VARCHAR(16) NOT NULL COMMENT '用户名',
    `password_hash` VARCHAR(255) NOT NULL COMMENT '加密密码',
    `email` VARCHAR(100) NOT NULL COMMENT '邮箱',
    `nickname` VARCHAR(16) NOT NULL COMMENT '昵称',
    `college_id` INT COMMENT '所属学院ID',
    `major_id` INT COMMENT '所属专业ID',
    `avatar_url` VARCHAR(255) COMMENT '头像URL',
    `reputation_score` INT NOT NULL DEFAULT 100 COMMENT '信誉分',
    `role_id` INT NOT NULL COMMENT '角色ID',
```

```
    `status` ENUM('active', 'locked', 'banned') NOT NULL DEFAULT 'active' COMMENT '账户状态',
    `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
    `updated_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '更新时间',
    PRIMARY KEY (`user_id`),
    UNIQUE KEY `uk_username` (`username`),
    UNIQUE KEY `uk_email` (`email`),
    KEY `fk_user_role` (`role_id`),
    KEY `fk_user_college` (`college_id`),
    KEY `fk_user_major` (`major_id`),
    CONSTRAINT `fk_user_role` FOREIGN KEY (`role_id`) REFERENCES `roles` (`role_id`),
    CONSTRAINT `fk_user_college` FOREIGN KEY (`college_id`) REFERENCES `colleges` (`college_id`),
    CONSTRAINT `fk_user_major` FOREIGN KEY (`major_id`) REFERENCES `majors` (`major_id`)
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COMMENT = '用户表';

```

roles表

sql

```sql
CREATE TABLE `roles` (
    `role_id` INT NOT NULL AUTO_INCREMENT COMMENT '角色ID',
    `role_name` VARCHAR(50) NOT NULL COMMENT '角色名称',
    `description` VARCHAR(500) COMMENT '角色描述',
    PRIMARY KEY (`role_id`),
    UNIQUE KEY `uk_role_name` (`role_name`)
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COMMENT = '角色表';

```

permissions表

sql

```sql
CREATE TABLE `permissions` (
    `permission_id` INT NOT NULL AUTO_INCREMENT COMMENT '权限ID',
    `permission_name` VARCHAR(50) NOT NULL COMMENT '权限名称',
    `description` VARCHAR(500) COMMENT '权限描述',
    PRIMARY KEY (`permission_id`),
    UNIQUE KEY `uk_permission_name` (`permission_name`)
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COMMENT = '权限表';

```

Role_Permissions表（角色 - 权限关联）

```sql
CREATE TABLE `role_permissions` (
    `role_id` INT NOT NULL COMMENT '角色ID',
    `permission_id` INT NOT NULL COMMENT '权限ID',
    PRIMARY KEY (`role_id`, `permission_id`),
    CONSTRAINT `fk_rp_role` FOREIGN KEY (`role_id`) REFERENCES `roles` (`role_id`) ON DELETE CASCADE,
    CONSTRAINT `fk_rp_permission` FOREIGN KEY (`permission_id`) REFERENCES `permissions` (`permission_id`) ON DELETE CASCADE
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COMMENT = '角色-权限关联表';
```

Colleges表

```sql
CREATE TABLE `colleges` (
    `college_id` INT NOT NULL AUTO_INCREMENT COMMENT '学院ID',
    `college_name` VARCHAR(50) NOT NULL COMMENT '学院名称',
    `school` VARCHAR(50) NOT NULL COMMENT '所属学校',
    PRIMARY KEY (`college_id`),
    UNIQUE KEY `uk_college_name` (`college_name`)
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COMMENT = '学院表';

```

Majors表

```sql
CREATE TABLE `majors` (
    `major_id` INT NOT NULL AUTO_INCREMENT COMMENT '专业ID',
    `major_name` VARCHAR(50) NOT NULL COMMENT '专业名称',
    `college_id` INT NOT NULL COMMENT '学院ID',
    PRIMARY KEY (`major_id`),
    KEY `fk_major_college` (`college_id`),
    CONSTRAINT `fk_major_college` FOREIGN KEY (`college_id`) REFERENCES `colleges` (`college_id`),
    UNIQUE KEY `uk_major_college` (`major_name`, `college_id`)
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COMMENT = '专业表';

```

Courses表

```sql
CREATE TABLE `courses` (
    `course_id` INT NOT NULL AUTO_INCREMENT COMMENT '课程ID',
    `name` VARCHAR(100) NOT NULL COMMENT '课程名称',
    `teacher_id` INT NOT NULL COMMENT '教师ID',
    `credit` DECIMAL(2, 1) NOT NULL COMMENT '学分',
    `major_id` INT NOT NULL COMMENT '专业ID',
    `grade` VARCHAR(20) NOT NULL COMMENT '年级',
    `description` VARCHAR(1000) COMMENT '课程简介',
    `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
    `updated_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '更新时间',
    PRIMARY KEY (`course_id`),
    KEY `fk_course_teacher` (`teacher_id`),
    KEY `fk_course_major` (`major_id`),
    KEY `idx_course_name` (`name`),
    CONSTRAINT `fk_course_teacher` FOREIGN KEY (`teacher_id`) REFERENCES `teachers` (`teacher_id`),
    CONSTRAINT `fk_course_major` FOREIGN KEY (`major_id`) REFERENCES `majors` (`major_id`)
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COMMENT = '课程表';

```

Teachers表

```sql
CREATE TABLE `teachers` (
    `teacher_id` INT NOT NULL AUTO_INCREMENT COMMENT '教师ID',
    `name` VARCHAR(50) NOT NULL COMMENT '教师姓名',
    `college_id` INT NOT NULL COMMENT '学院ID',
    `introduction` VARCHAR(1000) COMMENT '教师简介',
    PRIMARY KEY (`teacher_id`),
    KEY `fk_teacher_college` (`college_id`),
    CONSTRAINT `fk_teacher_college` FOREIGN KEY (`college_id`) REFERENCES `colleges` (`college_id`)
```

```sql
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COMMENT = '教师表';

```

Course_Comments表

```sql
CREATE TABLE `course_comments` (
    `comment_id` INT NOT NULL AUTO_INCREMENT COMMENT '评论ID',
    `user_id` INT NOT NULL COMMENT '用户ID',
    `course_id` INT NOT NULL COMMENT '课程ID',
    `content` TEXT NOT NULL COMMENT '评论内容',
    `parent_id` INT COMMENT '父评论ID',
    `likes` INT DEFAULT 0 COMMENT '点赞数',
    `is_visible` BOOLEAN DEFAULT TRUE COMMENT '是否显示',
    `status` ENUM('normal', 'deleted_by_user', 'deleted_by_admin') DEFAULT 'normal' COMMENT '评论状态',
    `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
    PRIMARY KEY (`comment_id`),
    KEY `fk_cc_user` (`user_id`),
    KEY `fk_cc_course` (`course_id`),
    KEY `fk_cc_parent` (`parent_id`),
    CONSTRAINT `fk_cc_user` FOREIGN KEY (`user_id`) REFERENCES `users` (`user_id`),
    CONSTRAINT `fk_cc_course` FOREIGN KEY (`course_id`) REFERENCES `courses` (`course_id`) ON DELETE CASCADE,
    CONSTRAINT `fk_cc_parent` FOREIGN KEY (`parent_id`) REFERENCES `course_comments` (`comment_id`) ON DELETE NO ACTION
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COMMENT = '课程评论表';

```

Course_Ratings表

```sql
CREATE TABLE `Course_Ratings` (
  `rating_id` INT PRIMARY KEY AUTO_INCREMENT,
  `user_id` INT,
```

```
    `course_id` INT,
    `recommendation` DECIMAL(2,1),
    `difficulty` INT,
    `workload` INT,
    `usefulness` INT,
    `is_visible` TINYINT(1),
    `created_at` TIMESTAMP NOT NULL
);
```

Resources表

sql

```sql
CREATE TABLE `resources` (
    `resource_id` INT NOT NULL AUTO_INCREMENT COMMENT '资源ID',
    `title` VARCHAR(255) NOT NULL COMMENT '资源标题',
    `description` VARCHAR(500) COMMENT '资源简介',
    `file_path` VARCHAR(255) NOT NULL COMMENT '文件路径',
    `file_type` ENUM('pdf', 'docx', 'pptx', 'zip') NOT NULL COMMENT '文件类型',
    `file_size` INT NOT NULL COMMENT '文件大小(字节)',
    `uploader_id` INT NOT NULL COMMENT '上传者ID',
    `course_id` INT NOT NULL COMMENT '课程ID',
    `download_count` INT DEFAULT 0 COMMENT '下载次数',
    `average_rating` DECIMAL(2, 1) DEFAULT 0 COMMENT '平均评分',
    `rating_count` INT DEFAULT 0 COMMENT '评分人数',
    `status` ENUM('normal', 'low_quality', 'pending_review') DEFAULT 'pending_review' COMMENT '状态',
    `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
    PRIMARY KEY (`resource_id`),
    KEY `fk_resource_uploader` (`uploader_id`),
    KEY `fk_resource_course` (`course_id`),
    KEY `idx_resource_status` (`status`),
    CONSTRAINT `fk_resource_uploader` FOREIGN KEY (`uploader_id`) REFERENCES `users` (`user_id`),
    CONSTRAINT `fk_resource_course` FOREIGN KEY (`course_id`) REFERENCES `courses` (`course_id`) ON DELETE CASCADE
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COMMENT = '资源表';

```

Resource_Ratings表

```sql
CREATE TABLE `course_ratings` (
    `rating_id` INT NOT NULL AUTO_INCREMENT COMMENT '评分ID',
    `user_id` INT NOT NULL COMMENT '用户ID',
    `course_id` INT NOT NULL COMMENT '课程ID',
    `recommendation` DECIMAL(2, 1) NOT NULL COMMENT '综合推荐度(0-5)',
    `difficulty` INT NOT NULL COMMENT '课程难度(1-5)',
    `workload` INT NOT NULL COMMENT '作业压力(1-5)',
    `usefulness` INT NOT NULL COMMENT '知识实用性(1-5)',
    `is_visible` BOOLEAN DEFAULT TRUE COMMENT '是否显示',
    `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
    PRIMARY KEY (`rating_id`),
    KEY `fk_cr_user` (`user_id`),
    KEY `fk_cr_course` (`course_id`),
    CONSTRAINT `fk_cr_user` FOREIGN KEY (`user_id`) REFERENCES `users` (`user_id`),
    CONSTRAINT `fk_cr_course` FOREIGN KEY (`course_id`) REFERENCES `courses` (`course_id`) ON DELETE CASCADE,
    UNIQUE KEY `uk_user_course` (`user_id`, `course_id`)
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COMMENT = '课程评分表';

```

Resource_Comments表

```sql
CREATE TABLE `resource_comments` (
    `comment_id` INT NOT NULL AUTO_INCREMENT COMMENT '评论ID',
    `user_id` INT NOT NULL COMMENT '用户ID',
    `resource_id` INT NOT NULL COMMENT '资源ID',
    `content` TEXT NOT NULL COMMENT '评论内容',
    `parent_id` INT COMMENT '父评论ID',
    `likes` INT DEFAULT 0 COMMENT '点赞数',
    `is_visible` BOOLEAN DEFAULT TRUE COMMENT '是否显示',
    `status` ENUM('normal', 'deleted_by_user', 'deleted_by_admin') DEFAULT 'normal' COMMENT '评论状态',
```

```sql
    `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
    PRIMARY KEY (`comment_id`),
    KEY `fk_rc_user` (`user_id`),
    KEY `fk_rc_resource` (`resource_id`),
    KEY `fk_rc_parent` (`parent_id`),
    CONSTRAINT `fk_rc_user` FOREIGN KEY (`user_id`) REFERENCES `users` (`user_id`),
    CONSTRAINT `fk_rc_resource` FOREIGN KEY (`resource_id`) REFERENCES `resources` (`resource_id`) ON DELETE CASCADE,
    CONSTRAINT `fk_rc_parent` FOREIGN KEY (`parent_id`) REFERENCES `resource_comments` (`comment_id`) ON DELETE NO ACTION
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COMMENT = '资源评论表';


```

Tags表

```sql
CREATE TABLE `tags` (
    `tag_id` INT NOT NULL AUTO_INCREMENT COMMENT '标签ID',
    `tag_name` VARCHAR(50) NOT NULL COMMENT '标签名称',
    PRIMARY KEY (`tag_id`),
    UNIQUE KEY `uk_tag_name` (`tag_name`)
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COMMENT = '标签表';

```

Resource_Tags表（资源 - 标签关联）

```sql
CREATE TABLE `resource_tags` (
    `resource_id` INT NOT NULL COMMENT '资源ID',
    `tag_id` INT NOT NULL COMMENT '标签ID',
    PRIMARY KEY (`resource_id`, `tag_id`),
    KEY `fk_rt_tag` (`tag_id`),
    CONSTRAINT `fk_rt_resource` FOREIGN KEY (`resource_id`) REFERENCES `resources` (`resource_id`) ON DELETE CASCADE,
```

```
    CONSTRAINT `fk_rt_tag` FOREIGN KEY (`tag_id`) REFERENCES `tags` (`tag_id`)
ON DELETE CASCADE
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COMMENT = '资源标签关联表';


```

Favorites表

sql

```sql
CREATE TABLE `favorites` (
    `favorite_id` INT NOT NULL AUTO_INCREMENT COMMENT '收藏ID',
    `user_id` INT NOT NULL COMMENT '用户ID',
    `target_id` INT NOT NULL COMMENT '对象ID',
    `target_type` ENUM('course', 'resource', 'teacher') NOT NULL COMMENT '对象类型',
    `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '收藏时间',
    PRIMARY KEY (`favorite_id`),
    KEY `fk_favorite_user` (`user_id`),
    KEY `idx_favorite_type` (`user_id`, `target_type`),
    CONSTRAINT `fk_favorite_user` FOREIGN KEY (`user_id`) REFERENCES `users` (`user_id`) ON DELETE CASCADE,
    UNIQUE KEY `uk_user_target` (`user_id`, `target_id`, `target_type`)
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COMMENT = '收藏表';

```

Reviews表

sql

```sql
CREATE TABLE `reviews` (
    `review_id` INT NOT NULL AUTO_INCREMENT COMMENT '审核ID',
    `target_id` INT NOT NULL COMMENT '对象ID',
    `target_type` ENUM(
        'resource',
        'course_rating',
        'resource_rating',
        'comment'
```

```
    ) NOT NULL COMMENT '对象类型',
    `reason` VARCHAR(500) NOT NULL COMMENT '审核原因',
    `status` ENUM('pending', 'approved', 'rejected') DEFAULT 'pending' COMMENT '状态',
    `priority` INT DEFAULT 3 COMMENT '优先级(1-5)',
    `reviewer_id` INT COMMENT '审核员ID',
    `reviewed_at` TIMESTAMP COMMENT '审核时间',
    `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
    PRIMARY KEY (`review_id`),
    KEY `fk_review_reviewer` (`reviewer_id`),
    KEY `idx_review_status` (`status`),
    KEY `idx_review_priority` (`priority`),
    CONSTRAINT `fk_review_reviewer` FOREIGN KEY (`reviewer_id`) REFERENCES `users` (`user_id`)
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COMMENT = '审核表';

```

Reputation_Records表

sql

```sql
CREATE TABLE `reputation_records` (
    `record_id` INT NOT NULL AUTO_INCREMENT COMMENT '记录ID',
    `user_id` INT NOT NULL COMMENT '用户ID',
    `change_score` INT NOT NULL COMMENT '分数变化',
    `reason` TEXT NOT NULL COMMENT '变化原因',
    `related_id` INT COMMENT '关联对象ID',
    `related_type` ENUM('resource', 'comment', 'rating') COMMENT '关联对象类型',
    `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '记录时间',
    PRIMARY KEY (`record_id`),
    KEY `fk_reprec_user` (`user_id`),
    KEY `idx_reputation_time` (`created_at`),
    CONSTRAINT `fk_reprec_user` FOREIGN KEY (`user_id`) REFERENCES `users` (`user_id`) ON DELETE CASCADE
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COMMENT = '信誉分记录表';

```

Items表

```sql
CREATE TABLE `items` (
    `item_id` INT NOT NULL AUTO_INCREMENT COMMENT '物品ID',
    `name` VARCHAR(100) NOT NULL COMMENT '物品名称',
    `type` ENUM(
        'avatar_frame',
        'background',
        'medal',
        'nickname_color'
    ) NOT NULL COMMENT '物品类型',
    `price` INT NOT NULL COMMENT '所需积分',
    `description` VARCHAR(500) COMMENT '物品描述',
    `image_url` VARCHAR(255) COMMENT '预览图URL',
    PRIMARY KEY (`item_id`),
    UNIQUE KEY `uk_item_name` (`name`)
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COMMENT = '物品表';

```

User_Items表

```sql
CREATE TABLE `user_items` (
    `user_item_id` INT NOT NULL AUTO_INCREMENT COMMENT '记录ID',
    `user_id` INT NOT NULL COMMENT '用户ID',
    `item_id` INT NOT NULL COMMENT '物品ID',
    `is_used` BOOLEAN DEFAULT FALSE COMMENT '是否启用',
    `obtained_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '获取时间',
    PRIMARY KEY (`user_item_id`),
    KEY `fk_ui_user` (`user_id`),
    KEY `fk_ui_item` (`item_id`),
    CONSTRAINT `fk_ui_user` FOREIGN KEY (`user_id`) REFERENCES `users` (`user_id`) ON DELETE CASCADE,
    CONSTRAINT `fk_ui_item` FOREIGN KEY (`item_id`) REFERENCES `items` (`item_id`) ON DELETE CASCADE,
    UNIQUE KEY `uk_user_item` (`user_id`, `item_id`)
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COMMENT = '用户物品表';
```

5.2. 视图设计

v_course_avg_rating（课程平均评分视图）

```sql
CREATE VIEW `v_course_avg_rating` AS
SELECT
    `course_id`,
    AVG(`recommendation`) AS `avg_recommendation`,
    AVG(`difficulty`) AS `avg_difficulty`,
    AVG(`workload`) AS `avg_workload`,
    AVG(`usefulness`) AS `avg_usefulness`,
    COUNT(*) AS `rating_count`
FROM `Course_Ratings`
WHERE `is_visible` = 1
GROUP BY `course_id`;
```

v_resource_avg_rating（资源平均评分视图）

```sql
CREATE VIEW `v_resource_avg_rating` AS
SELECT
    `resource_id`,
    AVG(`recommendation`) AS `avg_recommendation`,
    COUNT(*) AS `rating_count`
FROM `Resource_Ratings`
GROUP BY `resource_id`;
```

v_pending_reviews（待审核内容视图）

```sql
```

```sql
CREATE VIEW `v_pending_reviews` AS
SELECT
    `review_id`,
    `target_id`,
    `target_type`,
    `reason`,
    `priority`,
    `created_at`
FROM `Reviews`
WHERE `status` = 'pending'
ORDER BY `priority` DESC, `created_at` ASC;
```

5.3. 索引设计

用户表索引

```sql
CREATE INDEX `idx_users_username` ON `Users` (`username`);
CREATE INDEX `idx_users_email` ON `Users` (`email`);
CREATE INDEX `idx_users_college_id` ON `Users` (`college_id`);
CREATE INDEX `idx_users_major_id` ON `Users` (`major_id`);
CREATE INDEX `idx_users_role_id` ON `Users` (`role_id`);
CREATE INDEX `idx_users_status` ON `Users` (`status`);
```

课程相关索引

```sql
CREATE INDEX `idx_courses_teacher_id` ON `Courses` (`teacher_id`);
CREATE INDEX `idx_courses_major_id` ON `Courses` (`major_id`);
CREATE INDEX `idx_courses_grade` ON `Courses` (`grade`);
CREATE INDEX `idx_courses_created_at` ON `Courses` (`created_at`);
```

评论与评分索引

```sql
CREATE INDEX `idx_course_comments_user_id` ON `Course_Comments` (`user_id`);
CREATE INDEX `idx_course_comments_course_id` ON `Course_Comments` (`course_id`);
CREATE INDEX `idx_course_comments_status` ON `Course_Comments` (`status`);
CREATE INDEX `idx_course_ratings_course_id` ON `Course_Ratings` (`course_id`);
CREATE INDEX `idx_course_ratings_user_id` ON `Course_Ratings` (`user_id`);
```

资源相关索引

```sql
CREATE INDEX `idx_resources_uploader_id` ON `Resources` (`uploader_id`);
CREATE INDEX `idx_resources_course_id` ON `Resources` (`course_id`);
CREATE INDEX `idx_resources_status` ON `Resources` (`status`);
CREATE INDEX `idx_resources_created_at` ON `Resources` (`created_at`);
CREATE UNIQUE INDEX `uk_resources_title` ON `Resources` (`title`);
```

收藏与举报索引

```sql
CREATE INDEX `idx_favorites_user_id` ON `Favorites` (`user_id`);
CREATE INDEX `idx_favorites_target_type` ON `Favorites` (`target_type`);
CREATE INDEX `idx_reviews_target_type` ON `Reviews` (`target_type`);
CREATE INDEX `idx_reviews_status` ON `Reviews` (`status`);
```

5.4. 其他对象设计

5.4.1. 存储过程

sp_update_user_reputation（更新用户信誉分）

```sql
DELIMITER //
CREATE PROCEDURE `sp_update_user_reputation`(
    IN p_user_id INT,
    IN p_change_score INT,
    IN p_reason TEXT,
    IN p_related_id INT,
    IN p_related_type ENUM('course_comment', 'resource_comment', 'resource_upload', 'review_approved', 'violation')
)
BEGIN
    INSERT INTO `Reputation_Records` (`user_id`, `change_score`, `reason`, `related_id`, `related_type`, `created_at`)
    VALUES (p_user_id, p_change_score, p_reason, p_related_id, p_related_type, NOW());

    UPDATE `Users`
    SET `reputation_score` = `reputation_score` + p_change_score,
        `updated_at` = NOW()
    WHERE `user_id` = p_user_id;
END //
DELIMITER ;
```

sp_auto_hide_low_quality_resources（自动隐藏低质量资源）

```sql
DELIMITER //
CREATE PROCEDURE `sp_auto_hide_low_quality_resources`()
BEGIN
    UPDATE `Resources`
    SET `status` = 'disabled'
    WHERE `resource_id` IN (
        SELECT `resource_id` FROM `v_resource_avg_rating`
        WHERE `rating_count` > 10 AND `avg_recommendation` < 2.0
    )
    AND `status` = 'approved';
END //
DELIMITER ;
```

sp_calculate_resource_ratings （计算资源评分）

```sql
DELIMITER //
CREATE PROCEDURE `sp_calculate_resource_ratings`(IN p_resource_id INT)
BEGIN
    DECLARE avg_rating DECIMAL(2,1);
    DECLARE rating_count INT;

    SELECT
        AVG(`recommendation`),
        COUNT(*)
    INTO avg_rating, rating_count
    FROM `resource_ratings`
    WHERE `resource_id` = p_resource_id;

    UPDATE `resources`
    SET
        `average_rating` = COALESCE(avg_rating, 0),
        `rating_count` = rating_count
    WHERE `resource_id` = p_resource_id;
END //
DELIMITER ;
```

5.4.2. 触发器

tri_users_updated_at（用户表更新时间触发器）

```sql
DELIMITER //
CREATE TRIGGER `tri_users_updated_at`
BEFORE UPDATE ON `Users`
FOR EACH ROW
BEGIN
    SET NEW.`updated_at` = NOW();
```

```
END //
DELIMITER ;
```

tri_courses_updated_at（课程表更新时间触发器）

```sql
DELIMITER //
CREATE TRIGGER `tri_courses_updated_at`
BEFORE UPDATE ON `Courses`
FOR EACH ROW
BEGIN
    SET NEW.`updated_at` = NOW();
END //
DELIMITER ;
```

tri_resources_before_insert（资源插入前触发器）

```sql
DELIMITER //
CREATE TRIGGER `tri_resources_before_insert`
BEFORE INSERT ON `Resources`
FOR EACH ROW
BEGIN
    IF NEW.`status` IS NULL THEN
        SET NEW.`status` = 'pending_review';
    END IF;
END //
DELIMITER ;
```

六、安全设计

6.1. 用户角色与权限

6.1.1 角色定义

- 管理员：用户管理、内容审核、系统维护
- 普通用户：浏览课程、发表评论、上传资源等基本操作
- 受限用户：信誉分低于60分，限制评论、评分、上传功能

6.1.2 权限控制实现

- 通过 `Roles` 、 `Permissions` 、 `Role_Permissions` 表实现RBAC模型
- 后端接口进行权限验证，确保用户只能访问授权功能
- 前端根据用户角色动态显示菜单和功能按钮

6.2. 数据敏感性与加密

6.2.1 敏感数据加密

- 用户密码：采用bcrypt算法加密存储，加盐处理
- 用户邮箱：存储时进行单向哈希处理，用于找回密码验证
- 会话令牌：使用JWT令牌，设置合理过期时间

6.2.2 数据传输安全

- 所有API接口强制使用HTTPS协议
- 敏感操作（登录、支付、密码修改）要求二次验证

6.3. 审计

创建系统操作日志表

```sql
CREATE TABLE `System_Logs` (
    `log_id` INT PRIMARY KEY AUTO_INCREMENT,
    `user_id` INT,
    `action` VARCHAR(100) NOT NULL,
    `target_type` VARCHAR(50),
    `target_id` INT,
    `ip_address` VARCHAR(45),
    `user_agent` TEXT,
    `request_params` TEXT,
    `created_at` TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

七、数据库管理与维护说明

7.1. 备份策略

7.1.1 备份类型与频率

- 全量备份：每日凌晨2点执行，保留30天
- 增量备份：每6小时执行一次，保留7天
- 二进制日志备份：实时同步到备份服务器

7.1.2 备份存储

- 本地存储：最近3天备份，用于快速恢复
- 云端存储：所有备份同步到云存储，异地容灾
- 加密存储：所有备份文件进行AES-256加密

7.2. 优化策略

7.2.1 性能监控

- 启用MySQL慢查询日志，阈值设置为2秒
- 定期分析数据库连接数、锁等待、缓存命中率

7.2.2 容量规划

- 监控表空间使用情况，提前预警
- 每季度评估数据增长趋势，调整存储方案

7.2.3 索引维护

- 每月分析索引使用情况，删除无效索引
- 根据查询模式调整索引策略

八、附录

8.1. SQL初始化脚本

8.1.1 数据库创建脚本

代码块

```
-- 创建数据库，指定字符集和排序规则
CREATE DATABASE IF NOT EXISTS `zhihui_course_selection`
CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;
```

```
5
6   -- 切换到目标数据库
7   USE `zhihui_course_selection`;
```

8.1.2 基础数据初始化脚本

1. 角色表（roles）初始化

代码块

```
1   -- 插入系统预设角色
2   INSERT INTO `Roles` (`role_name`, `description`)
3   VALUES
4   ('admin', '系统管理员，拥有全部操作权限'),
5   ('normal_user', '普通用户，拥有浏览、评论、上传等基础权限'),
6   ('restricted_user', '受限用户，信誉分低于60分，限制部分功能');
```

2. 权限表（permissions）初始化

代码块

```
1   -- 插入系统预设权限
2   INSERT INTO `Permissions` (`permission_name`, `description`)
3   VALUES
4   -- 用户管理权限
5   ('user_management', '管理所有用户信息，包括查询、修改、禁用等'),
6   ('role_management', '管理角色信息，包括创建、修改、删除角色'),
7   ('permission_management', '管理权限信息，包括创建、修改、删除权限'),
8
9   -- 课程相关权限
10  ('course_view', '浏览课程信息'),
11  ('course_comment', '发表课程评论'),
12  ('course_rating', '对课程进行评分'),
13  ('course_management', '管理课程信息，包括创建、修改、删除课程'),
14
15  -- 资源相关权限
16  ('resource_view', '浏览课程资源'),
17  ('resource_upload', '上传课程资源'),
18  ('resource_comment', '发表资源评论'),
19  ('resource_rating', '对资源进行评分'),
20  ('resource_review', '审核用户上传的资源'),
21  ('resource_management', '管理资源信息，包括修改、删除资源'),
22
23  -- 收藏与举报权限
24  ('favorite_operate', '收藏课程、资源等内容'),
25  ('review_submit', '提交举报内容'),
```

```
('review_management', '处理举报内容，包括审核、标记状态'),

-- 信誉与物品权限
('reputation_view', '查看用户信誉分记录'),
('reputation_operate', '调整用户信誉分'),
('item_management', '管理虚拟物品信息'),
('item_exchange', '用户兑换虚拟物品'),

-- 系统维护权限
('system_log_view', '查看系统操作日志'),
('system_config', '修改系统配置参数');
```

3. 角色 - 权限关联表（Role_Permissions）初始化

代码块

```
-- 管理员角色关联全部权限
INSERT INTO `Role_Permissions` (`role_id`, `permission_id`)
SELECT 1, `permission_id` FROM `Permissions`;

-- 普通用户角色关联基础权限
INSERT INTO `Role_Permissions` (`role_id`, `permission_id`)
SELECT 2, `permission_id` FROM `Permissions`
WHERE `permission_name` IN (
    'course_view', 'course_comment', 'course_rating',
    'resource_view', 'resource_upload', 'resource_comment', 'resource_rating',
    'favorite_operate', 'review_submit', 'item_exchange'
);

-- 受限用户角色关联部分权限
INSERT INTO `Role_Permissions` (`role_id`, `permission_id`)
SELECT 3, `permission_id` FROM `Permissions`
WHERE `permission_name` IN (
    'course_view', 'resource_view', 'favorite_operate'
);
```

4. 学院表（Colleges）初始化

代码块

```
INSERT INTO `Colleges` (`college_name`, `school`)
VALUES
('计算机科学与技术学院', 'XX大学'),
('电子信息工程学院', 'XX大学'),
('工商管理学院', 'XX大学'),
('文学院', 'XX大学'),
```

```
('外国语学院', 'XX大学'),
('数理学院', 'XX大学');
```

5. 专业表（Majors）初始化

代码块

```sql
-- 计算机科学与技术学院专业
INSERT INTO `Majors` (`major_name`, `college_id`)
VALUES
('计算机科学与技术', 1),
('软件工程', 1),
('人工智能', 1),
('网络工程', 1);

-- 电子信息工程学院专业
INSERT INTO `Majors` (`major_name`, `college_id`)
VALUES
('电子信息工程', 2),
('通信工程', 2),
('自动化', 2);

-- 工商管理学院专业
INSERT INTO `Majors` (`major_name`, `college_id`)
VALUES
('工商管理', 3),
('市场营销', 3),
('会计学', 3);

-- 文学院专业
INSERT INTO `Majors` (`major_name`, `college_id`)
VALUES
('汉语言文学', 4),
('新闻学', 4),
('广告学', 4);

-- 外国语学院专业
INSERT INTO `Majors` (`major_name`, `college_id`)
VALUES
('英语', 5),
('日语', 5),
('德语', 5);

-- 数理学院专业
INSERT INTO `Majors` (`major_name`, `college_id`)
VALUES
```

```
('数学与应用数学', 6),
('物理学', 6),
('统计学', 6);
```

6. 虚拟物品表（Items）初始化

代码块

```
INSERT INTO `Items` (`name`, `type`, `description`, `image_url`)
VALUES
('星空头像框', 'avatar_frame', '带有星空特效的头像框，兑换需100信誉分',
'https://....'),
```

8.1.3 初始化说明

1. 执行顺序：先执行 “数据库创建脚本”，再依次执行 “基础数据初始化脚本” 中的各表插入语句。
2. 示例数据：学院、专业、虚拟物品等表的初始化数据为示例内容，实际部署时需根据项目所在高校的真实信息修改。
3. 权限调整：可根据实际业务需求，修改 `Role_Permissions` 表的插入数据，调整不同角色的权限范围。
4. 测试账号：若需创建测试用户，可在 `users` 表中插入测试数据（注意密码需使用 bcrypt 加密），示例如下：

代码块

```
-- 测试管理员账号（用户名：admin，密码：Admin123!，已通过bcrypt加密）
INSERT INTO `Users` (`username`, `password_hash`, `email`, `nickname`,
`role_id`, `status`, `created_at`, `updated_at`)
VALUES
('admin', '$2a$10$EixZaYbB.rK4fl8x2q7Me.2Q7wwt2ke20eF6H13C9s0hF6xG6u5aK',
'admin@zhihuixk.com', '系统管理员', 1, 'active', NOW(), NOW());

-- 测试普通用户账号（用户名：student1，密码：Student123!，已通过bcrypt加密）
INSERT INTO `Users` (`username`, `password_hash`, `email`, `nickname`,
`college_id`, `major_id`, `role_id`, `status`, `created_at`, `updated_at`)
VALUES
('student1', '$2a$10$9t0zX5Fw6G7h8J9k0L1m2n3O4p5Q6r7S8t9U0V1W2X3Y4Z5a6b7',
'student1@zhihuixk.com', '普通学生', 1, 1, 2, 'active', NOW(), NOW());
```