数据库设计说明书

福州大学 2022 级软件工程实践“研途无忧”小组

2024 年 10 月 30 日

目录

第一章　数据库设计

本章节详细介绍“福小研”考研辅导软件的数据库设计，包括编写目的、数据库策略以及命名规范。通过规范的数据库设计，确保数据的完整性、一致性和高效性，为系统的稳定运行和未来的扩展提供坚实的基础。

1.1　编写目的

本数据库设计文档旨在为“福小研”考研辅导软件提供全面、系统的数据管理方案。通过明确数据库的结构、策略和命名规范，确保数据存储的高效性、可靠性和可维护性。文档内容包括数据库策略、命名规范、实体关系模型（ER 图）、关系数据模型及对象关系映射（ORM），为开发团队提供详细的指导，促进团队协作和项目顺利实施。

1.2　数据库策略

数据库策略是确保数据库设计符合系统需求、性能要求及数据安全的重要组成部分。本节将介绍“福小研”软件在数据库设计中的关键策略，包括数据库对象长度、数据完整性、规范化设计与性能的权衡，以及字段类型的定义与使用策略。

1.2.1　数据库对象长度策略

为了确保数据库的高效性和数据的一致性，必须对数据库对象（如表名、字段名等）的长度进行合理的规划。

- **表名长度**：建议不超过 30 个字符，使用有意义的英文单词或缩写，避免使用特殊字符。
- **字段名长度**：建议不超过 50 个字符，采用描述性名称，避免过长或含糊不清的命名。
- **索引名长度**：索引名称应简洁明了，建议不超过 30 个字符，便于识别和管理。

1.2.2　数据完整性策略

数据完整性是确保数据库中数据的准确性和一致性的关键。为此，“福小研”软件在数据库设计中采用以下数据完整性策略：

- **实体完整性**：每个表必须有一个主键，确保每条记录的唯一性。

- **参照完整性**：通过外键约束，确保表之间的引用关系合法，防止“孤立”记录的出现。
- **域完整性**：为每个字段定义合适的数据类型和约束条件，如非空、唯一、默认值等，确保数据符合预期格式和范围。
- **用户定义完整性**：根据业务需求，设定特定的规则和触发器，确保数据操作符合业务逻辑。

1.2.3 规范化设计与性能之间的权衡策略

规范化设计通过消除数据冗余和依赖性，提升数据一致性。然而，过度规范化可能导致查询性能下降。为此，“福小研”软件在数据库设计中采取以下权衡策略：

- **第三范式（3NF）**：大部分表遵循第三范式，确保数据的高一致性和最小冗余。
- **适度反规范化**：针对高频查询和性能瓶颈的部分表，适度进行反规范化设计，减少联表操作，提高查询效率。
- **索引优化**：通过合理的索引设计，提升查询性能，尤其是在关联查询和大数据量操作时。
- **分区与分片**：对于极大数据量的表，采用分区或分片策略，分散数据存储，提升查询和写入性能。

1.2.4 字段类型的定义与使用策略

合理的数据类型定义不仅影响存储效率，还关系到数据操作的准确性和性能。

- **数值类型**：对于计数、金额等字段，选择合适的整数或浮点类型，避免不必要的精度损失。
- **字符串类型**：使用固定长度的 `CHAR` 类型存储固定格式的数据，使用 `VARCHAR` 存储可变长度的数据，避免浪费存储空间。
- **日期时间类型**：使用 `DATE`、`TIME`、`DATETIME` 等类型存储日期和时间数据，确保时间相关操作的准确性。
- **布尔类型**：使用 `BOOLEAN` 类型存储真/假值，简化逻辑判断。
- **枚举类型**：对于具有固定取值范围的字段，使用 `ENUM` 类型限制其取值，提升数据一致性。

在“福小研”考研辅导软件的数据库设计中，针对不同的集合（Collection）进行了详细的字段定义和使用策略。以下是主要集合的字段说明、MongoDB Schema 设计以及约束条件。

1.3 关系数据模型

描述数据库表结构，展示表之间的关系及其设计理由。

1.3.1 数据表结构与关系

“福小研”应用采用 uniCloud 云数据库，设计了多个数据表（集合）以支持不同的功能模块。以下是主要数据表及其关系：

数据表结构

数据表名	描述
users	存储用户信息，包括用户名、密码、个人简介、头像、福币等。
tasks	存储用户的待办事宜，包含任务内容、开始时间、结束时间、提醒设置等。
announcements	存储系统发布的通知公告，包含标题、内容、发布时间等。
study_plans	存储用户的学习计划，包含学习目标、科目、进度等。
resources	存储学习资源，如课程、题库、资料链接等。
posts	存储社区帖子，包含帖子内容、作者、评论、点赞等。
comments	存储帖子评论，包含评论内容、作者、回复对象等。
courses	存储课程信息，包含课程名称、课程链接、是否免费等。
leaderboard	存放用户的福币和更新信息，用于显示排行榜。
transactions	存放福币的支出收入交易记录。
dailytasks	存放每日任务的相关信息。

表之间的关系

- **用户（users）与任务（tasks）**：一对多关系。一个用户可以拥有多个待办任务。
- **用户（users）与帖子（posts）**：一对多关系。一个用户可以发布多个帖子。
- **帖子（posts）与评论（comments）**：一对多关系。一个帖子可以有多个评论。
- **课程（courses）与资源（resources）**：一对多关系。一个课程可以包含多个学习资源。
- **用户（users）与学习计划（study_plans）**：一对多关系。一个用户可以制定多个学习计划。
- **用户（users）与通知公告（announcements）**：多对多关系。多个用户可以接收多个通知公告，反之亦然。
- **用户（users）与排行榜（leaderboard）**：一对一关系。每个用户在排行榜中有唯一的记录。
- **用户（users）与交易记录（transactions）**：一对多关系。一个用户可以有多笔交易记录。
- **用户（users）与每日任务（dailytasks）**：一对多关系。一个用户可以有多个每日任务记录。

设计理由

- **规范化**：通过分离不同实体，减少数据冗余，确保数据的一致性和完整性。
- **扩展性**：清晰的关系设计允许未来功能的扩展和数据库的灵活扩展。
- **查询效率**：合理的表关系有助于优化查询性能，支持复杂的数据检索需求。

1.3.2 ER 图

第二章　对象关系映射

展示实体类和库表之间的映射关系，为开发提供直接的数据操作指南。

2.1　实体类与数据库表映射

实体类名	数据表名	主要属性映射
User	users	id ↔ _id, account ↔ account, password ↔ password, avatarUrl ↔ avatarUrl, nickname ↔ nickname, gender ↔ gender, year ↔ year, major ↔ major, school ↔ school, createdAt ↔ createdAt, updatedAt ↔ updatedAt
Task	tasks	id ↔ _id, content ↔ content, startTime ↔ startTime, endTime ↔ endTime, reminder ↔ reminder, userId ↔ userId
Announcement	announcements	id ↔ _id, title ↔ title, content ↔ content, publishDate ↔ publishDate
StudyPlan	study_plans	id ↔ _id, planName ↔ planName, status ↔ status, userId ↔ userId
Resource	resources	id ↔ _id, courseId ↔ courseId, resourceType ↔ resourceType, url ↔ url, description ↔ description
Post	posts	id ↔ _id, userAvatar ↔ userAvatar, userNickname ↔ userNickname, content ↔ content, likesCount ↔ likesCount, favoritesCount ↔ favoritesCount, sharesCount ↔ sharesCount, postDate ↔ postDate, hot ↔ hot
Comment	comments	id ↔ _id, postId ↔ postId, author ↔ author, content ↔ content, replyTo ↔ replyTo, commentDate ↔ commentDate

Course	courses	id ↔ _id, courseName ↔ courseName, courseLink ↔ courseLink, isFree ↔ isFree
Leaderboard	leaderboard	id ↔ _id, userId ↔ user_id, score ↔ score, updatedAt ↔ updatedAt
Transaction	transactions	id ↔ _id, userId ↔ user_id, description ↔ description, amount ↔ amount, date ↔ date
DailyTask	dailytasks	id ↔ _id, userId ↔ user_id, name ↔ name, reward ↔ reward, completed ↔ completed, date ↔ date

2.2 实体类定义示例

以下是部分实体类与数据库表的具体映射示例：

2.2.1 User 实体类与 users 数据表映射

Listing 2.1: User 实体类与 users 数据表映射

```
{
  "User": {
    "id": "ObjectId",
    "account": "String",
    "password": "String",
    "avatarUrl": "String",
    "nickname": "String",
    "gender": "String",
    "year": "String",
    "major": "String",
    "school": "String",
    "createdAt": "Long",
    "updatedAt": "Long"
  }
}
```

2.2.2 Post 实体类与 posts 数据表映射

Listing 2.2: Post 实体类与 posts 数据表映射

```
{
  "Post": {
    "id": "ObjectId",
    "userAvatar": "String",
    "userNickname": "String",
```

```
    "content": {
      "text": "String",
      "images": ["String"],
      "topics": ["String"]
    },
    "likesCount": "Int",
    "favoritesCount": "Int",
    "sharesCount": "Int",
    "postDate": "Date",
    "hot": "Boolean"
  }
}
```

2.2.3 Leaderboard 实体类与 leaderboard 数据表映射

Listing 2.3: Leaderboard 实体类与 leaderboard 数据表映射

```
{
  "Leaderboard": {
    "id": "ObjectId",
    "userId": "String",
    "score": "Int",
    "updatedAt": "Date"
  }
}
```

2.2.4 Transaction 实体类与 transactions 数据表映射

Listing 2.4: Transaction 实体类与 transactions 数据表映射

```
{
  "Transaction": {
    "id": "ObjectId",
    "userId": "String",
    "description": "String",
    "amount": "Int",
    "date": "String"
  }
}
```

2.2.5 DailyTask 实体类与 dailytasks 数据表映射

Listing 2.5: DailyTask 实体类与 dailytasks 数据表映射

```
{
  "DailyTask": {
    "id": "ObjectId",
    "userId": "String",
    "name": "String",
    "reward": "Int",
    "completed": "Boolean",
    "date": "String"
  }
}
```

2.3 ORM 映射关系图

2.3.1 映射说明

- 每个实体类对应一个数据库表，类的属性与表的字段一一对应。
- 一对多关系通过在多的一方添加外键（如 `userId` 在 `transactions` 和 `dailytasks` 表中）。
- 多对多关系通过中间表（如 `users_announcements`，未在本设计中列出，但可根据需要添加）。

通过清晰的对象关系映射，开发团队能够高效地进行数据库操作，实现业务逻辑与数据层的无缝连接。

第三章　字段类型的定义与使用策略

合理的数据类型定义不仅影响存储效率，还关系到数据操作的准确性和性能。根据“福小研”考研辅导软件的具体需求，以下是各类字段类型的定义与使用策略：

- **数值类型**：对于计数、金额等字段，选择合适的整数或浮点类型，避免不必要的精度损失。例如，`likesCount` 使用 `Int` 类型存储点赞人数。
- **字符串类型**：使用固定长度的 `CHAR` 类型存储固定格式的数据，如用户 ID，使用 `VARCHAR` 存储可变长度的数据，如用户名和描述，避免浪费存储空间。
- **日期时间类型**：使用 `DATE`、`TIME`、`DATETIME` 等类型存储日期和时间数据，如 `postDate` 和 `updatedAt`，确保时间相关操作的准确性。
- **布尔类型**：使用 `BOOLEAN` 类型存储真/假值，如 `hot` 和 `completed` 字段，简化逻辑判断。
- **枚举类型**：对于具有固定取值范围的字段，使用 `ENUM` 类型限制其取值，如 `status` 字段，仅接受 `未开始` 和 `已完成` 两种值，提升数据一致性。

3.1　数据库设计说明书

本数据库设计说明书详细介绍了“福小研”考研辅导软件的数据管理策略，包括各个数据库集合的设计细节。

3.1.1　社区帖子集合（posts）

字段说明

字段名称	数据类型	说明
_id	ObjectId	ID，系统自动生成，唯一标识每个帖子
userAvatar	String	用户头像的 URL
userNickname	String	用户昵称
content	Object	发帖内容（可选）
content.text	String	帖子文字内容（可选）
content.images	Array	帖子中包含的图片 URL 数组
content.topics	Array	帖子关联的话题标签（可选）

likesCount	Int	点赞人数，最小值为 0
favoritesCount	Int	收藏人数，最小值为 0
sharesCount	Int	转发人数，最小值为 0
postDate	Date	发帖日期
hot	Boolean	是否是热门帖子

MongoDB Schema

以下是社区帖子集合的 MongoDB Schema 设计：

Listing 3.1: 社区帖子集合 MongoDB Schema

```
{
  "bsonType": "object",
  "required": ["userAvatar", "userNickname", "postDate"],
  "properties": {
    "_id": {
      "bsonType": "objectId",
      "description": "ID，系统自动生成，唯一标识每个帖子"
    },
    "userAvatar": {
      "bsonType": "string",
      "description": "用户头像的URL"
    },
    "userNickname": {
      "bsonType": "string",
      "description": "用户昵称"
    },
    "content": {
      "bsonType": "object",
      "description": "发帖内容（可选）",
      "properties": {
        "text": {
          "bsonType": "string",
          "description": "帖子文字内容（可选）",
          "minLength": 0
        },
        "images": {
          "bsonType": "array",
          "items": {
            "bsonType": "string",
            "description": "图片的URL"
          },
          "description": "帖子中包含的图片URL数组",
          "minItems": 0
        },
```

```
        "topics": {
          "bsonType": "array",
          "items": {
            "bsonType": "string",
            "description": "关联的话题标签（可选）"
          },
          "description": "帖子关联的话题",
          "minItems": 0
        }
      }
    },
    "likesCount": {
      "bsonType": "int",
      "description": "点赞人数",
      "minimum": 0
    },
    "favoritesCount": {
      "bsonType": "int",
      "description": "收藏人数",
      "minimum": 0
    },
    "sharesCount": {
      "bsonType": "int",
      "description": "转发人数",
      "minimum": 0
    },
    "postDate": {
      "bsonType": "date",
      "description": "发帖日期"
    },
    "hot": {
      "bsonType": "boolean",
      "description": "是否是热门帖子"
    }
  }
}
```

约束条件

- **必填字段**：userAvatar、userNickname 和 postDate。
- **可选字段**：content（包含 text、images 和 topics）。
- **数值约束**：likesCount、favoritesCount 和 sharesCount 最小值为 0。
- **数据类型**：确保每个字段的数据类型符合定义，防止数据不一致。

3.1.2 课程集合（courses）

字段说明

字段名称	数据类型	说明
_id	ObjectId	ID，系统自动生成，唯一标识每个课程
courseName	String	课程名称
courseLink	String	课程链接
isFree	Boolean	是否免费（true 表示免费，false 表示收费）

MongoDB Schema

以下是课程集合的 MongoDB Schema 设计：

Listing 3.2: 课程集合 MongoDB Schema

```
{
  "bsonType": "object",
  "required": ["courseName", "courseLink", "isFree"],
  "properties": {
    "_id": {
      "bsonType": "objectId",
      "description": "ID，系统自动生成，唯一标识每个课程"
    },
    "courseName": {
      "bsonType": "string",
      "description": "课程名称"
    },
    "courseLink": {
      "bsonType": "string",
      "description": "课程链接"
    },
    "isFree": {
      "bsonType": "boolean",
      "description": "是否免费（true表示免费，false表示收费）"
    }
  }
}
```

约束条件

- **必填字段**：courseName、courseLink 和 isFree。
- **数据类型**：确保每个字段的数据类型符合定义，防止数据不一致。
- **值范围**：isFree 为布尔类型，只接受 true 或 false。

3.1.3 学习计划集合（study_plans）

字段说明

字段名称	数据类型	说明
_id	ObjectId	ID，系统自动生成，唯一标识每个学习计划
planName	String	学习计划具体名称
status	String	学习状态（未开始，已完成）

MongoDB Schema

以下是学习计划集合的 MongoDB Schema 设计：

Listing 3.3: 学习计划集合 MongoDB Schema

```
{
  "bsonType": "object",
  "required": ["planName", "status"],
  "properties": {
    "_id": {
      "bsonType": "objectId",
      "description": "ID，系统自动生成，唯一标识每个学习计划"
    },
    "planName": {
      "bsonType": "string",
      "description": "学习计划具体名称"
    },
    "status": {
      "bsonType": "string",
      "description": "学习状态（未开始，已完成）",
      "enum": ["未开始", "已完成"]
    }
  }
}
```

约束条件

- **必填字段**：planName 和 status。
- **值范围**：status 仅接受 未开始和 已完成两种值。
- **数据类型**：确保每个字段的数据类型符合定义，防止数据不一致。

3.1.4 排行榜集合（leaderboard）

排行榜数据库存放用户的福币和更新信息，通过用户 id 关联 users 数据库，获取头像和昵称等用户个人信息，显示在福榜上。

字段说明

字段名称	数据类型	说明
_id	ObjectId	ID，系统自动生成
user_id	String	用户 ID，关联 users 集合中的 _id
score	Int	用户的福币分数
updatedAt	Date	分数的最后更新时间

MongoDB Schema

以下是排行榜集合的 MongoDB Schema 设计：

Listing 3.4: 排行榜集合 MongoDB Schema

```
{
  "bsonType": "object",
  "required": [
    "user_id",
    "score",
    "updatedAt"
  ],
  "permission": {
    "read": true,
    "create": true,
    "update": true,
    "delete": false
  },
  "properties": {
    "_id": {
      "bsonType": "objectId",
      "description": "ID, 系统自动生成"
    },
    "user_id": {
      "bsonType": "string",
      "description": "用户ID, 关联 users 集合中的 _id"
    },
    "score": {
      "bsonType": "int",
      "description": "用户的福币分数"
    },
    "updatedAt": {
      "bsonType": "date",
      "description": "分数的最后更新时间"
    }
  }
}
```

约束条件

- **必填字段**：`user_id`、`score` 和 `updatedAt`。
- **数据类型**：确保每个字段的数据类型符合定义，防止数据不一致。
- **参照完整性**：`user_id` 必须关联到 `users` 集合中的有效 `_id`，确保数据的关联性。
- **权限控制**：`delete` 操作被禁止，仅允许读取、创建和更新。

3.1.5 交易记录集合（transactions）

交易记录数据库存放福币的支出收入交易记录。

字段说明

字段名称	数据类型	说明
_id	ObjectId	ID，系统自动生成
user_id	String	用户的唯一 ID，关联用户表
description	String	交易描述，如’签到奖励’、’兑换商品’
amount	Int	交易金额，正数表示收入，负数表示支出
date	String	交易日期，格式如’2020.9.19’

MongoDB Schema

以下是交易记录集合的 MongoDB Schema 设计：

Listing 3.5: 交易记录集合 MongoDB Schema

```
{
  "bsonType": "object",
  "required": [
    "user_id",
    "description",
    "amount",
    "date"
  ],
  "permission": {
    "read": true,
    "create": true,
    "update": false,
    "delete": false
  },
  "properties": {
    "_id": {
      "bsonType": "objectId",
      "description": "ID, 系统自动生成"
```

```
    },
    "user_id": {
      "bsonType": "string",
      "description": "用户的唯一ID，关联用户表"
    },
    "description": {
      "bsonType": "string",
      "description": "交易描述，如'签到奖励'、'兑换商品'"
    },
    "amount": {
      "bsonType": "int",
      "description": "交易金额，正数表示收入，负数表示支出"
    },
    "date": {
      "bsonType": "string",
      "description": "交易日期，格式如␣'2020.9.19'"
    }
  }
}
```

约束条件

- **必填字段**：user_id、description、amount 和 date。
- **数据类型**：确保每个字段的数据类型符合定义，防止数据不一致。
- **值范围**：amount 为整数，正数表示收入，负数表示支出。
- **日期格式**：date 字段应遵循'YYYY.MM.DD' 格式，确保日期的一致性和可解析性。
- **权限控制**：仅允许读取和创建，不允许更新和删除，以保护交易记录的完整性。

3.1.6 每日任务集合（dailytasks）

每日任务数据库存放每日任务的相关信息。

字段说明

字段名称	数据类型	说明
_id	ObjectId	ID，系统自动生成
user_id	String	用户的唯一 ID，用于区分不同用户的任务
name	String	任务名称，如' 签到打卡'
reward	Int	任务奖励数量
completed	Boolean	任务完成状态，true 表示已完成，false 表示未完成

date	String	任务日期，用于区分每日任务

MongoDB Schema

以下是每日任务集合的 MongoDB Schema 设计：

Listing 3.6: 每日任务集合 MongoDB Schema

```
{
  "bsonType": "object",
  "required": [
    "user_id",
    "name",
    "reward",
    "completed",
    "date"
  ],
  "permission": {
    "read": true,
    "create": true,
    "update": true,
    "delete": false
  },
  "properties": {
    "_id": {
      "bsonType": "objectId",
      "description": "ID，系统自动生成"
    },
    "user_id": {
      "bsonType": "string",
      "description": "用户的唯一ID，用于区分不同用户的任务"
    },
    "name": {
      "bsonType": "string",
      "description": "任务名称，如'签到打卡'"
    },
    "reward": {
      "bsonType": "int",
      "description": "任务奖励数量"
    },
    "completed": {
      "bsonType": "bool",
      "description": "任务完成状态，true表示已完成，false表示未完成"
    },
    "date": {
      "bsonType": "string",
```

```
      "description": "任务日期，用于区分每日任务"
    }
  }
}
```

约束条件

- **必填字段**：user_id、name、reward、completed 和 date。
- **数据类型**：确保每个字段的数据类型符合定义，防止数据不一致。
- **值范围**：
  - reward 为整数，表示任务奖励数量。
  - completed 为布尔类型，只接受 true 或 false。
  - date 字段应遵循'YYYY.MM.DD' 格式，确保日期的一致性和可解析性。
- **权限控制**：delete 操作被禁止，仅允许读取、创建和更新，确保每日任务记录的完整性。

3.2 命名规范

命名规范是数据库设计中的重要组成部分，良好的命名规范有助于提高数据库的可读性、可维护性和团队协作效率。本节将介绍“福小研”软件在数据库命名方面的具体规范。

3.2.1 数据库命名规则

- 使用小写字母和下划线分隔单词，避免使用大写字母和特殊字符。
- 表名、字段名应具有描述性，能够反映其存储的数据含义。
- 避免使用缩写，除非是公认的行业标准缩写。

3.2.2 数据库对象命名的一般原则

- **一致性**：所有数据库对象的命名应遵循统一的规则，确保一致性。
- **简洁性**：名称应简洁明了，避免过长或冗余。
- **可读性**：名称应易于理解和记忆，便于团队成员快速识别其含义。

3.2.3 表空间 (Tablespace) 命名规则

- 表空间名称应反映其存储的数据库对象类别或功能模块。
- 使用前缀或后缀标识表空间的用途，如 ts_users 表示用户相关表空间。

3.2.4 表 (Table) 命名规则

- 表名使用复数形式，如 `users`、`tasks`。
- 表名应准确描述存储的数据内容。
- 避免使用保留字或系统关键字作为表名。

3.2.5 字段命名规则

- 字段名应使用小写字母和下划线分隔单词，如 `user_id`、`created_at`。
- 主键字段统一命名为 `id`，外键字段命名为 `<referenced_table>_id`，如 `user_id`。
- 避免使用过于宽泛或模糊的名称，如 `data`、`info`。

3.2.6 视图 (View) 命名规则

- 视图名称应以 `vw_` 或 `view_` 作为前缀，如 `vw_user_tasks`。
- 名称应反映视图所展示的数据内容和来源。
- 避免使用复杂的命名结构，保持名称简洁。

3.2.7 序列 (Sequence) 命名规则

- 序列名称应以 `seq_` 作为前缀，并包含相关表名，如 `seq_users_id`。
- 确保序列名称的唯一性，避免与其他数据库对象名称冲突。

3.2.8 存储过程 (Procedure) 的命名规则

- 存储过程名称应以 `sp_` 作为前缀，如 `sp_create_user`。
- 名称应描述存储过程的功能和操作，如 `sp_update_task_status`。
- 使用动词开头，表明存储过程执行的动作。

3.2.9 函数 (Function) 的命名规则

- 函数名称应以 `fn_` 作为前缀，如 `fn_calculate_score`。
- 名称应描述函数的功能和返回结果，如 `fn_get_user_details`。
- 保持名称简洁，避免使用冗长的描述。

3.2.10 索引 (Index) 命名规范

- 索引名称应以 idx_ 作为前缀，后跟表名和字段名，如 idx_users_account。
- 对于复合索引，使用字段名的组合，如 idx_posts_user_id_post_date。
- 保持索引名称简洁明了，避免使用不必要的单词。

3.2.11 约束 (Constraint) 命名规范

- 约束名称应以 chk_(检查约束)、fk_(外键约束)、pk_(主键约束)等前缀标识,如 fk_tasks_user_id。
- 约束名称应包含相关表名和字段名，如 chk_users_age。
- 保持约束名称简洁，避免使用冗长或复杂的命名结构。