# 武汉大学资源与环境科学学院答题纸

年级 2024 学号 2024302181194 姓名 王李明

专业 计算机类（国家网络安全学院） 课程名称 数据库技术基础与应用开发

| 成绩 | 总分 | 一 | 二 | 三 | 四 | 五 | 六 | 七 | 八 | 九 | 十 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | |

## 基于 Go 语言与 MySQL 的汇率资讯交流平台后端设计与实现

一、 软件作品成果（50 分）

（一）作品简介

本项目名为“蓝鼠兑换”，是一个集汇率查询与社区交流于一体的信息管理系统后端。系统采用 RESTful API 架构风格，前端（Web 端）通过 HTTP 请求与后端交互。数据存储核心采用开源的 MySQL 数据库系统，利用 Go 语言的高并发特性和 GORM 库的便捷性，实现了高效的数据读写与事务处理。

（二）核心功能模块（4 个）

用户管理模块：支持用户注册、登录。

文章社区模块：用户可以发布关于财经、汇率分析的文章，支持文章的增删改查。

汇率数据模块：提供基础货币与目标货币的实时汇率查询、历史汇率记录。

点赞模块：支持用户对文章进行点赞，实时统计点赞数。（该模块不涉及 MySQL）

（三）数据库设计

系统核心依赖 MySQL 数据库，设计了以下关键数据表：

users：存储用户信息（用户名、加密密码）。

articles：存储预览内容、文章内容、作者关联、发布时间等。

exchange_rates：存储货币对、汇率值、更新时间等。

（四）成果展示

详见该文档同一文件夹下的 Exhibition 文件夹。

二、 成果描述（50 分）

（一）作品所用环境

开发环境与工具：

操作系统：Windows 11

编程语言：Go (Golang) 1.25.0

Web 框架：Gin (github.com/gin-gonic/gin)

数据库管理系统：MySQL 9.4

ORM 框架：GORM (gorm.io/gorm)

数据库可视化工具：DataGrip

开发工具 (IDE)：Visual Studio Code

（二）实现过程

1.**数据定义语言（DDL）**的映射与约束

系统主要包含三个核心实体，通过结构体标签（Tags）定义数据库约束。

① 用户模型 (models/user.go)

Gorm 代码如下：

```go
type User struct {
    gorm.Model
    Username string `gorm:"unique"` // 设置唯一索引，防止用户名重复
    Password string
    // 建立一对多关系：一个用户可以拥有多篇文章
    Articles []Article `json:"articles" gorm:"foreignKey:UserID"`
}
```

```go
type Model struct {
    ID        uint `gorm:"primarykey"`
    CreatedAt time.Time
    UpdatedAt time.Time
    DeletedAt DeletedAt `gorm:"index"`
}
```

分析：嵌入 gorm.Model 自动生成了 id （主键） 及 deleted_at （软删除字段）。gorm:"unique" 标签在数据库层面为 Username 创建了唯一索引，防止用户名重复。

对应 MySQL 语句如下：

```sql
CREATE TABLE `users` (
  `id` bigint unsigned NOT NULL AUTO_INCREMENT,
  `created_at` datetime(3) DEFAULT NULL,
  `updated_at` datetime(3) DEFAULT NULL,
  `deleted_at` datetime(3) DEFAULT NULL,
  `username` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `idx_users_username` (`username`),
  KEY `idx_users_deleted_at` (`deleted_at`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

② 文章模型（models/article.go）

Gorm 代码如下：

```go
type Article struct {
    gorm.Model
    Title   string `binding:"required"`
    Content string `binding:"required"`
    Preview string `binding:"required"`
    // 外键定义
    UserID uint `json:"userId"`
    User   User `json:"-" gorm:"foreignKey:UserID"` // Belongs To 关系
}
```

MySQL 语句略。

分析：Gorm 用嵌入结构体的方式来定义外键关系。UserID 作为外键，存储关联用户的 ID，确立了 Article 属于 User 的归属关系。

③ 汇率模型（models/exchange_rate.go）

Gorm 代码如下：

```go
type ExchangeRate struct {
    ID           uint      `gorm:"primarykey" json:"_id"`
    FromCurrency string    `json:"fromCurrency" binding:"required" gorm:"index:idx_currency_pair"`
    ToCurrency   string    `json:"toCurrency" binding:"required" gorm:"index:idx_currency_pair"`
    Rate         float64   `json:"rate" binding:"required"`
    Date         time.Time `json:"date"`
}
```

MySQL 语句略。

分析：在FromCurrency 和ToCurrency上使用相同的索引名idx_currency_pair，GORM 会生成如下 SQL 创建复合索引，极大提升 WHERE from_currency = ? AND to_currency = ? 的查询速度：

```sql
CREATE INDEX `idx_currency_pair` ON `exchange_rates` (`from_currency`, `to_currency`);
```

2.**数据操作语言（DML）**的实现与 Hook 机制

插入前的数据处理

在 models/user.go 中，使用了 BeforeSave 钩子来处理密码加密。

Gorm 代码如下：

```go
func (u *User) BeforeSave(tx *gorm.DB) (err error) {
    if len(u.Password) > 0 {
        u.Password, _ = utils.HashPassword(u.Password) // 加密
    }
    return
}
```

当 auth_controller.go 调用 global.Db.Create(&user)、向 MySQL 数据库中创建并保存新用户时，系统先执行 Hook，再执行如下 SQL：

```sql
INSERT INTO `users` (`created_at`,`updated_at`,`deleted_at`,`username`,`password`)
VALUES ('2025-12-04...','...',NULL,'testuser','hashed_secret_pwd');
```

3.**数据查询语言（DQL）**的映射

在 auth_controller.go 的登录逻辑中，Gorm 代码如下：

```go
global.Db.Where("username = ?", input.Username).First(&user)
```

对应 SQL 语句如下：

```sql
SELECT * FROM `users`
WHERE username = 'input_user'
AND `users`.`deleted_at` IS NULL  -- 自动过滤已软删除的数据
ORDER BY `users`.`id`
LIMIT 1;
```

4.**事务控制**的应用

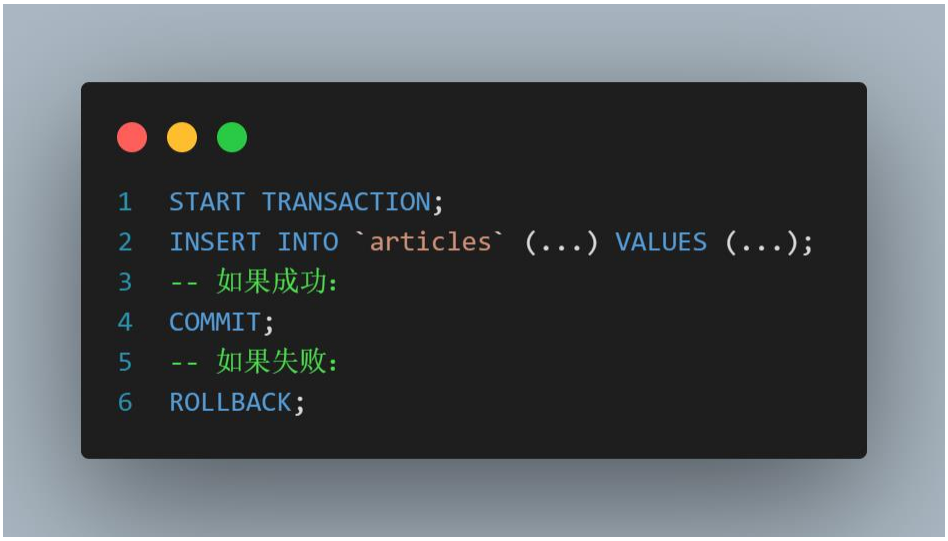
在 controllers/article_controller.go 的 CreateArticle 方法中，为了确保文章创建过程的原子性（例如未来可能扩展的同时更新统计表），我使用了显式的事务控制。Gorm 代码如下：

```go
// 1. 开启事务
tx := global.Db.Begin()

// 2. 执行数据库操作
if err := tx.Create(&article).Error; err != nil {
    tx.Rollback() // 出错则回滚
    return
}

// 3. 提交事务
tx.Commit()
```

对应 SQL 语句如下：



```sql
START TRANSACTION;
INSERT INTO `articles` (...) VALUES (...);
-- 如果成功:
COMMIT;
-- 如果失败:
ROLLBACK;
```

（三） 遇到的主要难点及克服方法

难点 1：复合索引的定义

问题：如何通过 GORM 定义跨多个字段的索引以优化汇率查询？

解决：查阅文档后，发现通过在不同字段 tag 中指定相同的 index 名称（如 idx_currency_pair），即可让 GORM 生成复合索引。

难点 2：业务逻辑与数据库操作的分治

问题：密码加密逻辑如果写在 Controller 层，会导致代码冗余且容易遗漏。

解决：利用 GORM 的 BeforeSave Hook，将加密逻辑下沉到 Model 层，确保无论何处调用创建用户，密码都会被自动加密。

（四）心得

通过本次课程设计，我深刻理解了 ORM 框架并非是 SQL 的替代品，而是 SQL 的封装。

SQL 是核心：无论 GORM 语法多么简洁，开发者必须清楚其背后生成的 SQL 语句（如 First 隐含的 ORDER BY 和 LIMIT），才能避免性能陷阱。

规范化设计：数据库的 E-R 建模是项目成功的基石，合理的外键和索引设计能显著提升系统的健壮性。

这次实践让我从理论走向了应用，掌握了构建一个基于 MySQL 的现代化后端系统的完整流程。