** 江苏大学京江学院**

数据库

课程设计报告

学院： 京江学院

班级：

姓名：

学号：

2024年 6 月

一、 引言   
1．设计目的. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .3  
2．设计要求 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3   
3. 设计环境 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

二、概要设计   
1．系统需求分析. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .3   
2. 系统功能结构. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .3   
3. 概念模型设计. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .4

三、详细设计   
1．数据库设计. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .4  
2．完整性设计. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .6

3. 安全性设计. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .8  
4．主要功能模块设计. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .8

四、调试与运行 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19

五、课设小结 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22

六、参考文献. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .22

附录源码

一、引言

1.设计目的

为了更好的管理江苏大学运动会，此管理系统可以通过系统查询自己的比赛项目及成绩，裁判员可以记录运动员成绩并进行统计，管理员可以管理比赛项目、学院和参赛运动员的信息。该系统提高了运动会管理的效率和准确性。

2.设计要求

运动员功能：查看所在学院的名称、领队、总积分、总名次，查看自己参加的比赛项目、成绩、积分

裁判员功能：记录运动员比赛成绩，查看运动员比赛成绩、积分、名次，查看各学院的总积分、总名次

管理员功能：管理比赛项目（增加、删除、修改），管理学院信息，管理参赛运动员信息

使用对象许可和命令许可、角色控制设计安全性控制检查程序

为每一参照关系设计插入、删除、修改触发器

保证数据的完整性和安全性，通过主键、外键和CHECK约束实现数据完整性控制

实现应用程序设计、编程、优化功能

3. 设计环境

编程语言：Java

数据库：MySQL

开发工具：IntelliJ IDEA

操作系统：Windows 11

二、概要设计

1. 系统需求分析

系统需要满足以下需求：

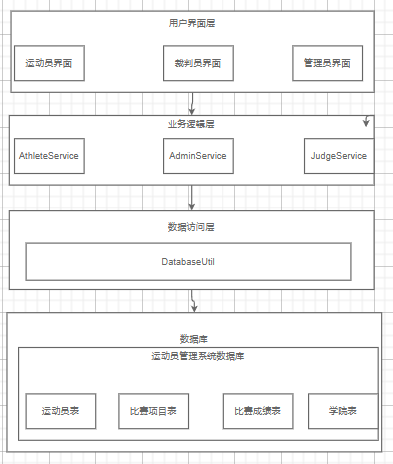
运动员：查询自己所在学院的名称、领队、本学院的总积分、总名次；查询自己参加的比赛项目、每个项目的成绩和积分。

裁判员：记录运动员参加比赛项目的成绩；查看运动员参加比赛项目的成绩、积分和名次；查看各学院的总积分和总名次。

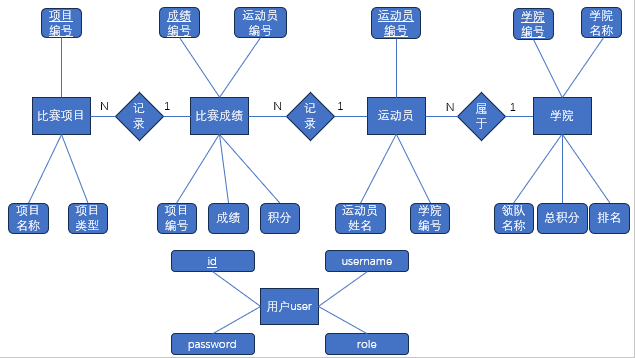
管理员：管理（增加、删除、修改）比赛项目；管理学院信息；管理每个学院的参赛运动员信息。

2.系统功能结构

系统结构图如图

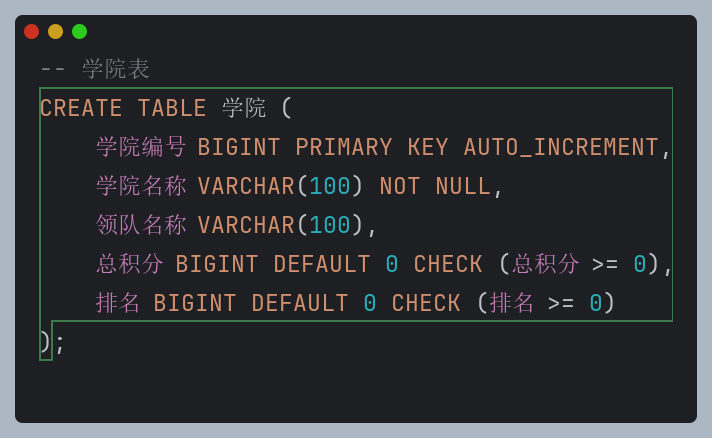


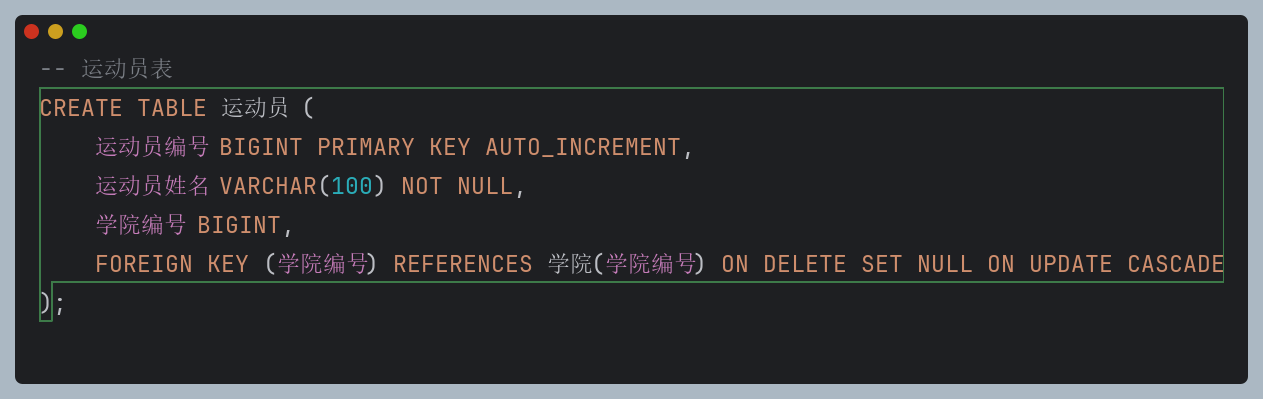
3.概念模型设计

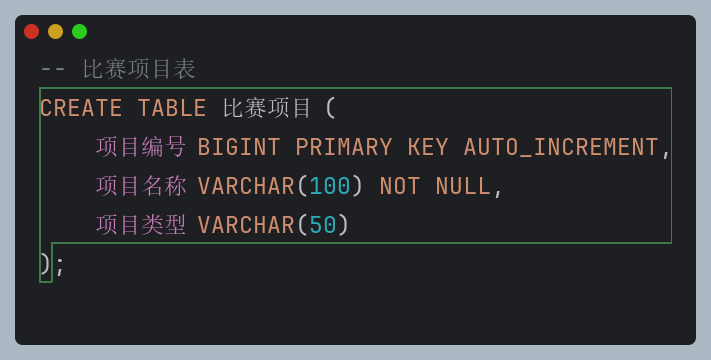
ER图如图所示

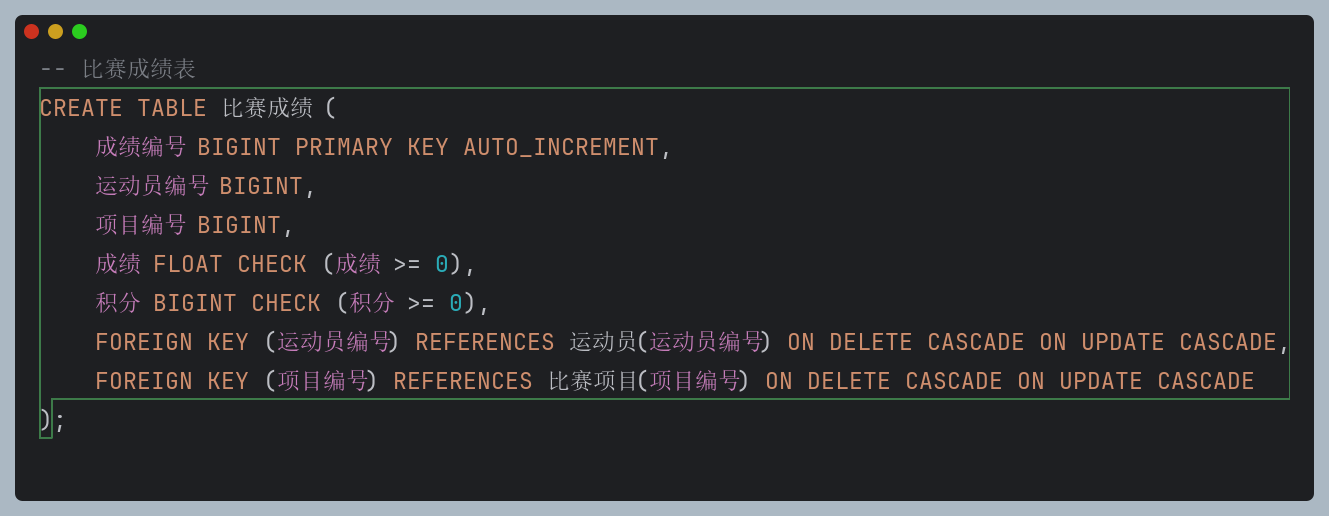
三、详细设计

1.数据库设计

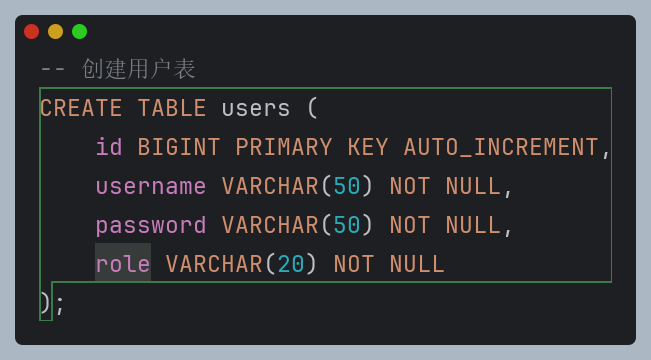
创建学院表

创建运动员表

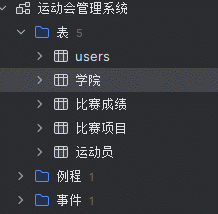
创建比赛项目表

创建比赛成绩表

创建用户表



第一范式分析：要求在满足所有关系模式的属性值都是原子值，没有重复的组。在这四个表中每一列都是原子的（如学院表的学院编号、学院名称、领队名称、总积分、排名），所以满足第一范式。

第二范式分析：要求在满足第一范式的基础上，消除非主属性对主键的部分依赖。具体来说，对于一个复合主键的表，非主属性必须完全依赖于整个主键，而不是部分依赖于其中的某一个部分。所有表都只有一个主键（非复合主键），因此所有非主属性都完全依赖于主键，并且不存在部分依赖的情况。（如学院编号是主键，学院名称、领队名称、总积分、排名都完全依赖于学院编号。所以满足第二范式。

第三范式分析：要求在满足第二范式的基础上，消除非主属性对其他非主属性的传递依赖。具体来说，一个表中的非主属性不能依赖于其他非主属性，必须直接依赖于主键。所有表的非主属性都直接依赖于主键，并且没有非主属性依赖于其他非主属性。（如学院编号是主键。学院名称、领队名称、总积分、排名都完全依赖于学院编号。没有非主属性依赖于其他非主属性）。所以满足第三范式。

综上，关系模式符合第三范式。

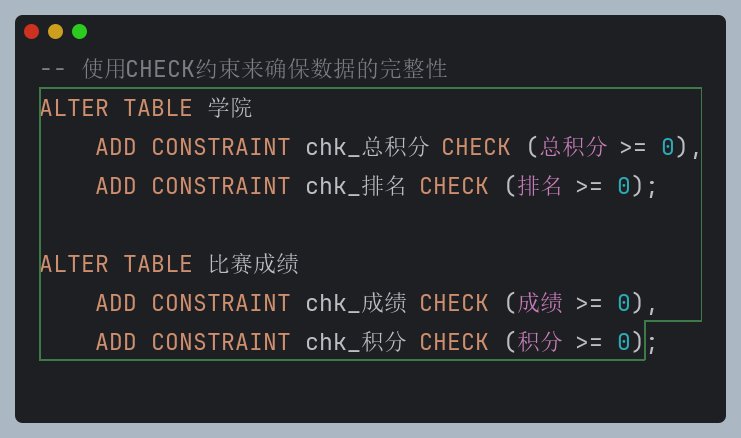
2．完整性设计

通过设计关系的主键约束、外键约束和CHECK约束，实现数据完整性控制

主键约束：每个表都有一个主键，确保记录的唯一性

外键约束：引用其他表的主键，确保参照完整性

CHECK约束：确保字段值符合特定条件



外键约束：确保数据的引用完整性。

CHECK约束：确保数据的有效性，例如成绩和积分必须为非负值。

触发器设计

学院表的触发器：

当删除学院时，设置相关运动员的 学院编号 为 NULL。

运动员表的触发器：

当删除运动员时，删除相关的比赛成绩记录。



比赛项目表的触发器：

当删除比赛项目时，删除相关的比赛成绩记录。



3.安全性设计

用户身份验证：通过用户名和密码进行登录验证。

权限控制：根据用户角色分配不同的操作权限（管理员、运动员、裁判员）。

4.主要功能模块设计

数据库模块：完成该系统核心任务。

用户登录模块：实现用户登录功能，根据用户角色显示不同的操作菜单。

管理员模块：可以管理（增加、删除、修改）比赛项目；可以管理学院信息；可以管理每个学院的参赛运动员信息。

裁判员模块：记录运动员参加比赛项目的成绩；查看运动员参加比赛项目的成绩、积分、名次；查看各学院的总积分、总名次等。

运动员模块：查看运动员所在学院的名称、领队、本学院的总积分、总名次；查看运动员参加比赛的项目、每个项目的成绩、积分等。

数据库MYSQL设计:

CREATE DATABASE 运动会管理系统;

USE 运动会管理系统;

-- 学院表

CREATE TABLE 学院 (

学院编号 BIGINT PRIMARY KEY AUTO\_INCREMENT,

学院名称 VARCHAR(100) NOT NULL,

领队名称 VARCHAR(100),

总积分 BIGINT DEFAULT 0 CHECK (总积分 >= 0),

排名 BIGINT DEFAULT 0 CHECK (排名 >= 0)

);

-- 运动员表

CREATE TABLE 运动员 (

运动员编号 BIGINT PRIMARY KEY AUTO\_INCREMENT,

运动员姓名 VARCHAR(100) NOT NULL,

学院编号 BIGINT,

FOREIGN KEY (学院编号) REFERENCES 学院(学院编号) ON DELETE SET NULL ON UPDATE CASCADE

);

-- 比赛项目表

CREATE TABLE 比赛项目 (

项目编号 BIGINT PRIMARY KEY AUTO\_INCREMENT,

项目名称 VARCHAR(100) NOT NULL,

项目类型 VARCHAR(50)

);

-- 比赛成绩表

CREATE TABLE 比赛成绩 (

成绩编号 BIGINT PRIMARY KEY AUTO\_INCREMENT,

运动员编号 BIGINT,

项目编号 BIGINT,

成绩 FLOAT CHECK (成绩 >= 0),

积分 BIGINT CHECK (积分 >= 0),

FOREIGN KEY (运动员编号) REFERENCES 运动员(运动员编号) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (项目编号) REFERENCES 比赛项目(项目编号) ON DELETE CASCADE ON UPDATE CASCADE

);

-- 创建用户表

CREATE TABLE users (

id BIGINT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(50) NOT NULL,

password VARCHAR(50) NOT NULL,

role VARCHAR(20) NOT NULL

);

-- 插入示例用户

INSERT INTO users (username, password, role) VALUES ('admin', 'admin\_password', 'admin');

INSERT INTO users (username, password, role) VALUES ('athlete', 'athlete\_password', 'athlete');

INSERT INTO users (username, password, role) VALUES ('judge', 'judge\_password', 'judge');

-- 创建用户和分配权限

CREATE USER 'athlete'@'localhost' IDENTIFIED BY 'athlete\_password';

GRANT SELECT ON 运动会管理系统.\* TO 'athlete'@'localhost';

CREATE USER 'judge'@'localhost' IDENTIFIED BY 'judge\_password';

GRANT SELECT, INSERT ON 运动会管理系统.\* TO 'judge'@'localhost';

CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin\_password';

GRANT ALL PRIVILEGES ON 运动会管理系统.\* TO 'admin'@'localhost';

FLUSH PRIVILEGES;

-- 使用CHECK约束来确保数据的完整性

ALTER TABLE 学院

ADD CONSTRAINT chk\_总积分 CHECK (总积分 >= 0),

ADD CONSTRAINT chk\_排名 CHECK (排名 >= 0);

ALTER TABLE 比赛成绩

ADD CONSTRAINT chk\_成绩 CHECK (成绩 >= 0),

ADD CONSTRAINT chk\_积分 CHECK (积分 >= 0);

-- 创建存储过程

DELIMITER //

CREATE PROCEDURE UpdateCollegeRanking()

BEGIN

-- 使用用户定义变量来重置排名

SET @rank := 0;

-- 重置所有学院的排名

UPDATE 学院

SET 排名 = 0

WHERE 学院编号 IS NOT NULL;

-- 更新学院排名

UPDATE 学院 a

JOIN (

SELECT 学院编号, (@rank := @rank + 1) AS 排名

FROM 学院

ORDER BY 总积分 DESC

) b ON a.学院编号 = b.学院编号

SET a.排名 = b.排名

WHERE a.学院编号 = b.学院编号;

END //

DELIMITER ;

-- 创建事件调度器定期执行存储过程

DELIMITER //

CREATE EVENT UpdateRankingEvent

ON SCHEDULE EVERY 1 DAY

DO

BEGIN

CALL UpdateCollegeRanking();

END //

DELIMITER ;

-- 创建触发器

DELIMITER //

-- 删除学院触发器

CREATE TRIGGER trg\_delete\_college

BEFORE DELETE ON 学院

FOR EACH ROW

BEGIN

UPDATE 运动员 SET 学院编号 = NULL WHERE 学院编号 = OLD.学院编号;

END //

-- 删除运动员触发器

CREATE TRIGGER trg\_delete\_athlete

BEFORE DELETE ON 运动员

FOR EACH ROW

BEGIN

DELETE FROM 比赛成绩 WHERE 运动员编号 = OLD.运动员编号;

END //

-- 删除比赛项目触发器

CREATE TRIGGER trg\_delete\_event

BEFORE DELETE ON 比赛项目

FOR EACH ROW

BEGIN

DELETE FROM 比赛成绩 WHERE 项目编号 = OLD.项目编号;

END //

-- 插入成绩时更新学院积分和排名

CREATE TRIGGER trg\_insert\_performance

AFTER INSERT ON 比赛成绩

FOR EACH ROW

BEGIN

-- 更新学院的总积分

UPDATE 学院

SET 总积分 = 总积分 + NEW.积分

WHERE 学院编号 = (SELECT 学院编号 FROM 运动员 WHERE 运动员编号 = NEW.运动员编号);

-- 更新学院排名

CALL UpdateCollegeRanking();

END //

-- 更新成绩时更新学院积分和排名

CREATE TRIGGER trg\_update\_performance

BEFORE UPDATE ON 比赛成绩

FOR EACH ROW

BEGIN

DECLARE old\_points BIGINT;

DECLARE new\_points BIGINT;

SET old\_points = OLD.积分;

SET new\_points = NEW.积分;

IF old\_points <> new\_points THEN

UPDATE 学院

SET 总积分 = 总积分 - old\_points + new\_points

WHERE 学院编号 = (SELECT 学院编号 FROM 运动员 WHERE 运动员编号 = OLD.运动员编号);

END IF;

-- 更新学院排名

CALL UpdateCollegeRanking();

END //

-- 删除成绩时更新学院积分和排名

CREATE TRIGGER trg\_delete\_performance

BEFORE DELETE ON 比赛成绩

FOR EACH ROW

BEGIN

UPDATE 学院

SET 总积分 = 总积分 - OLD.积分

WHERE 学院编号 = (SELECT 学院编号 FROM 运动员 WHERE 运动员编号 = OLD.运动员编号);

-- 更新学院排名

CALL UpdateCollegeRanking();

END //

DELIMITER ;

用户登录模块

public static String login(String username, String password) throws SQLException {

String query = "SELECT role FROM users WHERE username = ? AND password = ?";

try (Connection connection = DatabaseUtil.getConnection("admin");

PreparedStatement statement = connection.prepareStatement(query)) {

statement.setString(1, username);

statement.setString(2, password);

ResultSet resultSet = statement.executeQuery();

if (resultSet.next()) {

return resultSet.getString("role");

}

}

return null;

}

管理员模块

public class AdminService {

private static final String ROLE = "admin";

// 增加学院

public void addCollege(String collegeName, String leaderName) throws SQLException {

String query = "INSERT INTO 学院 (学院名称, 领队名称) VALUES (?, ?)";

try (Connection connection = DatabaseUtil.getConnection(ROLE);

PreparedStatement statement = connection.prepareStatement(query)) {

statement.setString(1, collegeName);

statement.setString(2, leaderName);

statement.executeUpdate();

System.out.println("学院添加成功");

}

}

// 增加比赛项目

public void addEvent(String eventName, String eventType) throws SQLException {

String query = "INSERT INTO 比赛项目 (项目名称, 项目类型) VALUES (?, ?)";

try (Connection connection = DatabaseUtil.getConnection(ROLE);

PreparedStatement statement = connection.prepareStatement(query)) {

statement.setString(1, eventName);

statement.setString(2, eventType);

statement.executeUpdate();

System.out.println("比赛项目添加成功");

}

}

// 删除比赛项目

public void deleteEvent(long eventId) throws SQLException {

String query = "DELETE FROM 比赛项目 WHERE 项目编号 = ?";

try (Connection connection = DatabaseUtil.getConnection(ROLE);

PreparedStatement statement = connection.prepareStatement(query)) {

statement.setLong(1, eventId);

statement.executeUpdate();

System.out.println("比赛项目删除成功");

}

}

// 修改比赛项目

public void updateEvent(long eventId, String eventName, String eventType) throws SQLException {

String query = "UPDATE 比赛项目 SET 项目名称 = ?, 项目类型 = ? WHERE 项目编号 = ?";

try (Connection connection = DatabaseUtil.getConnection(ROLE);

PreparedStatement statement = connection.prepareStatement(query)) {

statement.setString(1, eventName);

statement.setString(2, eventType);

statement.setLong(3, eventId);

statement.executeUpdate();

System.out.println("比赛项目更新成功");

}

}

// 管理学院信息

public void manageCollege(long collegeId, String collegeName, String leaderName) throws SQLException {

String query = "UPDATE 学院 SET 学院名称 = ?, 领队名称 = ? WHERE 学院编号 = ?";

try (Connection connection = DatabaseUtil.getConnection(ROLE);

PreparedStatement statement = connection.prepareStatement(query)) {

statement.setString(1, collegeName);

statement.setString(2, leaderName);

statement.setLong(3, collegeId);

statement.executeUpdate();

System.out.println("学院信息更新成功");

}

}

// 管理学院运动员信息

public void manageAthlete(long athleteId, String athleteName, long collegeId) throws SQLException {

String query = "UPDATE 运动员 SET 运动员姓名 = ?, 学院编号 = ? WHERE 运动员编号 = ?";

try (Connection connection = DatabaseUtil.getConnection(ROLE);

PreparedStatement statement = connection.prepareStatement(query)) {

statement.setString(1, athleteName);

statement.setLong(2, collegeId);

statement.setLong(3, athleteId);

statement.executeUpdate();

System.out.println("运动员信息更新成功");

}

}

// 添加运动员

public void addAthlete(long athleteId, String athleteName, long collegeId) throws SQLException {

String query = "INSERT INTO 运动员 (运动员编号, 运动员姓名, 学院编号) VALUES (?, ?, ?)";

try (Connection connection = DatabaseUtil.getConnection(ROLE);

PreparedStatement statement = connection.prepareStatement(query)) {

statement.setLong(1, athleteId);

statement.setString(2, athleteName);

statement.setLong(3, collegeId);

int rowsInserted = statement.executeUpdate();

if (rowsInserted > 0) {

System.out.println("运动员添加成功");

} else {

System.out.println("运动员添加失败");

}

}

}

// 删除运动员

public void deleteAthlete(long athleteId) throws SQLException {

// 先删除与该运动员相关的所有成绩记录

String deleteScoresQuery = "DELETE FROM 比赛成绩 WHERE 运动员编号 = ?";

try (Connection connection = DatabaseUtil.getConnection(ROLE);

PreparedStatement deleteScoresStatement = connection.prepareStatement(deleteScoresQuery)) {

deleteScoresStatement.setLong(1, athleteId);

deleteScoresStatement.executeUpdate();

}

// 再删除运动员

String deleteAthleteQuery = "DELETE FROM 运动员 WHERE 运动员编号 = ?";

try (Connection connection = DatabaseUtil.getConnection(ROLE);

PreparedStatement deleteAthleteStatement = connection.prepareStatement(deleteAthleteQuery)) {

deleteAthleteStatement.setLong(1, athleteId);

deleteAthleteStatement.executeUpdate();

System.out.println("运动员删除成功");

}

}

// 删除学院

public void deleteCollege(long collegeId) throws SQLException {

// 先删除与该学院相关的所有运动员及其成绩记录

String selectAthletesQuery = "SELECT 运动员编号 FROM 运动员 WHERE 学院编号 = ?";

try (Connection connection = DatabaseUtil.getConnection(ROLE);

PreparedStatement selectAthletesStatement = connection.prepareStatement(selectAthletesQuery)) {

selectAthletesStatement.setLong(1, collegeId);

ResultSet resultSet = selectAthletesStatement.executeQuery();

while (resultSet.next()) {

long athleteId = resultSet.getLong("运动员编号");

deleteAthlete(athleteId);

}

}

// 再删除学院

String deleteCollegeQuery = "DELETE FROM 学院 WHERE 学院编号 = ?";

try (Connection connection = DatabaseUtil.getConnection(ROLE);

PreparedStatement deleteCollegeStatement = connection.prepareStatement(deleteCollegeQuery)) {

deleteCollegeStatement.setLong(1, collegeId);

deleteCollegeStatement.executeUpdate();

System.out.println("学院删除成功");

}

}

}

运动员模块

public class AthleteService {

private static final String ROLE = "athlete";

// 查看运动员所在学院的信息

public void viewCollegeInfo(long athleteId) throws SQLException {

String query = "SELECT c.学院名称, c.领队名称, c.总积分, c.排名 " +

"FROM 运动员 a JOIN 学院 c ON a.学院编号 = c.学院编号 " +

"WHERE a.运动员编号 = ?";

try (Connection connection = DatabaseUtil.getConnection(ROLE);

PreparedStatement statement = connection.prepareStatement(query)) {

statement.setLong(1, athleteId);

ResultSet resultSet = statement.executeQuery();

if (resultSet.next()) {

System.out.println("学院名称: " + resultSet.getString("学院名称"));

System.out.println("领队名称: " + resultSet.getString("领队名称"));

System.out.println("总积分: " + resultSet.getInt("总积分"));

System.out.println("排名: " + resultSet.getInt("排名"));

}

}

}

// 查看运动员参加的比赛项目及成绩

public void viewAthleteEvents(long athleteId) throws SQLException {

String query = "SELECT e.项目名称, p.成绩, p.积分 " +

"FROM 比赛成绩 p JOIN 比赛项目 e ON p.项目编号 = e.项目编号 " +

"WHERE p.运动员编号 = ?";

try (Connection connection = DatabaseUtil.getConnection(ROLE);

PreparedStatement statement = connection.prepareStatement(query)) {

statement.setLong(1, athleteId);

ResultSet resultSet = statement.executeQuery();

while (resultSet.next()) {

System.out.println("项目名称: " + resultSet.getString("项目名称"));

System.out.println("成绩: " + resultSet.getFloat("成绩"));

System.out.println("积分: " + resultSet.getInt("积分"));

}

}

}

}

裁判员模块

public class JudgeService {

private static final String ROLE = "judge";

// 记录运动员成绩

public void recordPerformance(long athleteId, long eventId, float score, int points) throws SQLException {

if (!isAthleteExists(athleteId)) {

System.out.println("运动员ID不存在");

return;

}

if (!isEventExists(eventId)) {

System.out.println("比赛项目ID不存在");

return;

}

String query = "INSERT INTO 比赛成绩 (运动员编号, 项目编号, 成绩, 积分) VALUES (?, ?, ?, ?)";

try (Connection connection = DatabaseUtil.getConnection(ROLE);

PreparedStatement statement = connection.prepareStatement(query)) {

statement.setLong(1, athleteId);

statement.setLong(2, eventId);

statement.setFloat(3, score);

statement.setInt(4, points);

statement.executeUpdate();

System.out.println("成绩记录成功");

}

}

// 查看运动员比赛成绩

public void viewAthletePerformance(long athleteId) throws SQLException {

String query = "SELECT e.项目名称, p.成绩, p.积分 " +

"FROM 比赛成绩 p JOIN 比赛项目 e ON p.项目编号 = e.项目编号 " +

"WHERE p.运动员编号 = ?";

try (Connection connection = DatabaseUtil.getConnection(ROLE);

PreparedStatement statement = connection.prepareStatement(query)) {

statement.setLong(1, athleteId);

ResultSet resultSet = statement.executeQuery();

while (resultSet.next()) {

System.out.println("项目名称: " + resultSet.getString("项目名称"));

System.out.println("成绩: " + resultSet.getFloat("成绩"));

System.out.println("积分: " + resultSet.getInt("积分"));

}

}

}

// 查看各学院总积分和名次

public void viewCollegeScores() throws SQLException {

String query = "SELECT 学院名称, 总积分, 排名 FROM 学院 ORDER BY 排名";

try (Connection connection = DatabaseUtil.getConnection(ROLE);

PreparedStatement statement = connection.prepareStatement(query);

ResultSet resultSet = statement.executeQuery()) {

while (resultSet.next()) {

System.out.println("学院名称: " + resultSet.getString("学院名称"));

System.out.println("总积分: " + resultSet.getInt("总积分"));

System.out.println("排名: " + resultSet.getInt("排名"));

}

}

}

// 检查运动员是否存在

private boolean isAthleteExists(long athleteId) throws SQLException {

String query = "SELECT 1 FROM 运动员 WHERE 运动员编号 = ?";

try (Connection connection = DatabaseUtil.getConnection(ROLE);

PreparedStatement statement = connection.prepareStatement(query)) {

statement.setLong(1, athleteId);

ResultSet resultSet = statement.executeQuery();

return resultSet.next();

}

}

// 检查比赛项目是否存在

private boolean isEventExists(long eventId) throws SQLException {

String query = "SELECT 1 FROM 比赛项目 WHERE 项目编号 = ?";

try (Connection connection = DatabaseUtil.getConnection(ROLE);

PreparedStatement statement = connection.prepareStatement(query)) {

statement.setLong(1, eventId);

ResultSet resultSet = statement.executeQuery();

return resultSet.next();

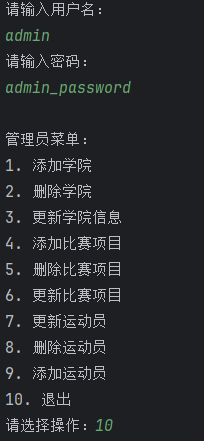
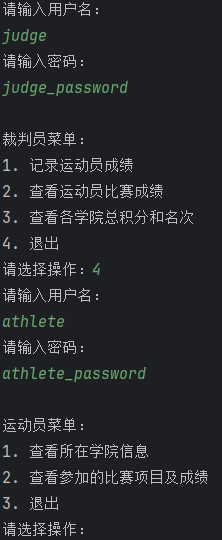
}

}

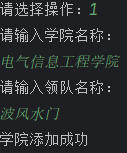
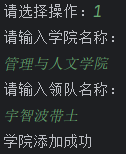
}

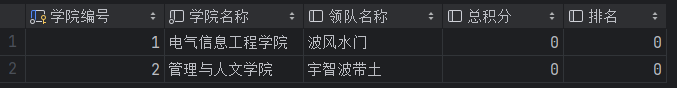
四、调试与运行

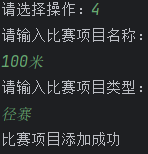
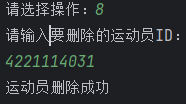
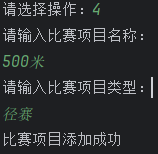
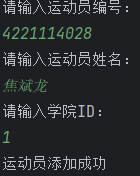
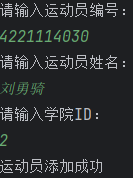
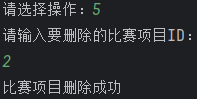
各用户的登录

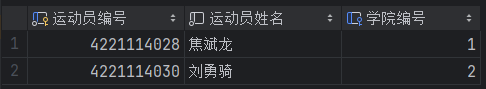


管理员管理数据



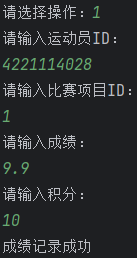
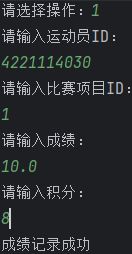


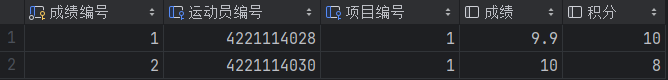
    



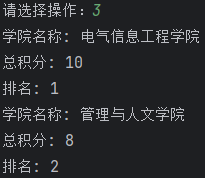
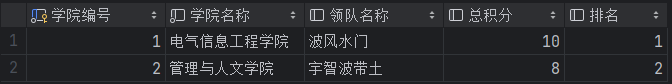


裁判员操作数据

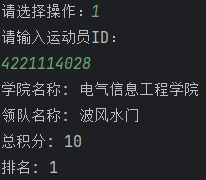
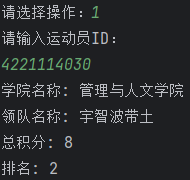


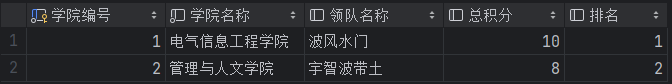


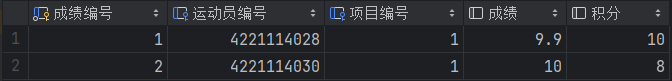


运动员查看数据









五、课设小结

这学期才学习完JAVA和数据库，于是我使用JAVA来实现各用户的功能，但是由于我能力薄弱，写这个课设的时候出现了许多错误，不过好在有互联网查询和CSDN、Github等众多社区帮我解决了难题。例如在实现管理员删除运动员这个功能时，我直接把运动员删除了，结果IDEA运行报错，在网上查询后才知道先要删除与该运动员相关的所有成绩记录才能再删除运动员，因为为了确保数据的完整性和参照完整性，数据库中的外键约束要求在删除主记录（即运动员）之前，必须先处理与之相关的从记录（即比赛成绩），比赛成绩表中的运动员编号是一个外键，引用了运动员表中的运动员编号。外键约束确保了比赛成绩记录中的运动员编号必须存在于运动员表中。如果直接删除运动员，数据库会违反外键约束，因为比赛成绩表中的相关记录会引用不存在的运动员。通过先删除相关的成绩记录，可以确保数据库中没有孤立的引用，保持数据的一致性和完整性。还有就是进行排名的时候无法排名以及运动员的积分无法计入学院总积分，通过CSDN才解决了问题。还有我在给学院进行排名的时候，我在想如何让学院自动排名更新而不需要人工干预呢？经过网上查询和CSDN后我知道了可以利用通过创建存储过程 UpdateCollegeRanking 实现学院排名的更新逻辑，并通过事件调度器 UpdateRankingEvent 定期（每天）调用该存储过程，来达到让学院的排名信息始终是最新的。以及在编写JAVA代码时，在我写各种方法的时候一直提示和标红“无法解析xxx表”，经过查询是未在IDEA里配置SQL方言问题，配置为Generic SQL后就不会提示标红了。

通过这个课程设计，我不仅掌握了数据库设计和SQL编程的基本技能，还培养了解决实际问题的能力。虽然过程中遇到了许多困难，但通过不断学习和实践，我逐步克服了这些挑战。这次课程设计经验对我的学习和成长有着重要的意义，我将继续努力，提升自己的技术水平。

六、参考文献

[1]数据库系统概论 –-高等教育出版社 王珊、萨师煊

[2]JAVA从入门到精通 –清华大学出版社 明日科技

[3]SQL成绩统计分析—排名问题 --CSDN No\_Bug\_

[4]数据库之触发器详解 --CSDN ~青萍之末~

[5]MySQL中的CHECK约束的使用方法 --CSDN ThDeBug

[6] Java连接数据库实现用户登录和注册功能 –CSDN 程序猿老李头

[7] mysql创建定时任务执行存储过程 –51CTO博客 mob649e81593bda

附录源码