实验题目	数据库设计与应用开发			实验日期	2022/5/1
班级	1903102	学号	1190200122	姓名	袁野

CS33503 数据库系统实验

实验检查记录

实验结果的正确性(60%)		表达能力(10%)	
实验过程的规范性(10%)		实验报告(20%)	
加分(5%)		总成绩(100%)	

实验报告

一、实验目的（介绍实验目的）

1. 学会正确运用概念数据库设计方法，正确使用实体-联系图(ER 图)表示概念数据模型。
2. 学会正确运用逻辑数据库设计方法，在概念数据模型的基础上，设计规范的关系数据库模式。
3. 学会正确运用物理数据库设计方法，根据工作负载，合理设计与调优数据库的存取方法与模式。
4. 掌握一种关系数据库管理系统(RDBMS)的使用方法，使用 SQL 创建、更新和查询关系数据库。
5. 掌握数据库系统应用开发方法。

二、实验环境（介绍实验使用的硬件设备、软件系统、开发工具等）

1、硬件设备：
i7-9750H CPU@2.60GHz 2.59GHz；1.8GHz；16G RAM;
2、软件系统
Ubuntu 20.04.4 LTS (GNU/Linux 5.10.102.1-microsoft-standard-WSL2 x86_64)
3、开发工具
MySQL

三、实验过程（介绍实验过程、设计方案、实现方法、实验结果等）

3.1 需求分析

在本次实验中，我编写了一个竞赛成绩管理系统，在该系统中，我们需要管理若干若干选手、队伍、比赛及相关成绩等信息，具体来说，该数据库需要维护的信息包括以下几部分：人员信息（学生编号，姓名，性别，邮箱，年级，学校，学院），队伍信息（队伍编号，队伍名字，队伍队员、教练），比赛（比赛编号，比赛名字，比赛日期，比赛级别，承办学校），学校信息（学校编号，名字，所在国家、省、市）以及具体参赛信息（队伍、比赛、名次、奖牌、队伍类别）。

该系统支持随时注册新队员、教练，添加新学校，注册新比赛，注册新队伍以及添加具体成绩信息等，同时支持根据某些具体信息对上述内容进行筛选查询，其中，除了支持根据

输入内容精确查找以外，还支持输入关键字进行模糊搜索。以上部分将在下面具体提到。

3.2 概念数据库设计

在该系统中，概念数据库的实体型包括人员、队伍、学校、比赛，联系型包括队伍成员关系，队伍在比赛中的具体参赛信息，成员所属学校关系，队伍所属学校关系，比赛承办学校关系。

- 人员：编号，姓名，性别，邮箱，年级（职位），学校，学院；
- 队伍：编号，名字
- 比赛：编号，名字，日期，级别
- 学校：编号，名字，国家、省、市
- 队伍成员关系：成员，队伍，角色
- 具体参赛信息：队伍，比赛，名次，奖牌，队伍类别
- 成员所属学校：成员，学校
- 队伍所属学校：队伍，学校
- 比赛承办学校：比赛，学校

根据上述关系我们可以画出 ER 图：

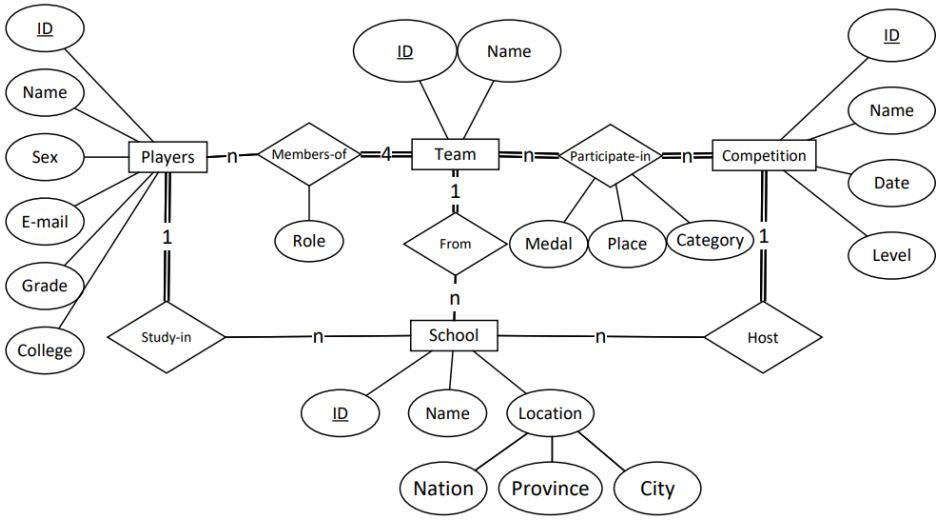

图 1 概念数据库对应的 ER 图

3.3 逻辑数据库设计

由于人员所属学校关系、队伍所属学校关系，比赛承办学校关系中，每个人员、队伍、比赛所包含的这些关系只有一个，因此我将这些信息分别储存在人员、队伍、比赛中。

实体型转换结果如下：

- 人员：Players(ID,Name,Sex,E-mail,School,College,Grade)
- 队伍：Team(ID,Name,School)
- 比赛：Competition(ID,Name,Time,Level,Host)
- 学校：School(ID,Name,Nation,Province,City)

关系型转换结果如下：

- 人员所属学校关系：Members-of(Player,Team,Role)
- 具体参赛信息：Participate-in(Team,Competition,Medal,Place,Category)

3.4 物理数据库设计

对于所有的实体型，均有 ID 作为主索引，关系型 Members-of 以(Player,Team)二元组作为主键，Participate-in 以(Team,Competition)二元组作为主键，MySql 会以这些主键建立主索引。此外，根据实际情况，我还对成员的邮箱信息、队伍名字、学校名字、比赛名字分别建立了四个唯一索引。为了保持数据的一致性，在这些表中涉及其他表的列均设计成外键。

在功能设计中，有一些查询功能，比如根据人员 ID 查询该人员的参赛历史信息，即 Participate-in 中的信息，则需要使用语句

```
SELECT * FROM participate_in  WHERE Team IN (SELECT Team FROM members_of WHERE Player = /*输入人员 ID*/)
```

在同一场比赛中，不能有两个不同的队伍包含同一个队员，因此需要执行下面的 sql 语句来查找该比赛中其他队伍是否与输入的这支队伍有队员交集。

```
SELECT * FROM (SELECT * FROM members_of WHERE Team in (SELECT Team FROM participate_in WHERE Competition = /*输入比赛名字*/)) AS P1 INNER JOIN (SELECT * FROM members_of WHERE Team = /*输入该队名*/ AND Role != '教练') AS P2 ON P1.Player = P2.Player ;
```

在列出所有学校时，需要列出当前学校已经注册的成员数，如果直接在 School 表中记录的话不符合范式要求，因此采用以下查询语句：

```
SELECT school.Id, school.Name, school.Nation, school.Province, school.City, IFNULL(P.num, 0) FROM school LEFT JOIN (SELECT school, count(*) FROM players GROUP BY school) AS P(school, num) ON school.Name = P.School
```

其余的操作就是比较基础的增删查改操作。

3.5 数据库建立

在本次实验中，我使用 MySQL 来建立数据库。其对应的 MySQL 代码为

```
1.  CREATE DATABASE  IF NOT EXISTS `competition_manager`;
2.  USE `competition_manager`;
3.
4.  CREATE TABLE `competition` (
5.    `ID` int NOT NULL AUTO_INCREMENT,
6.    `Name` varchar(45) DEFAULT NULL,
7.    `Time` date DEFAULT NULL,
8.    `Level` enum('国际级','国家级','地区级','省级','市级') DEFAULT NULL,
9.    `Host` varchar(45) DEFAULT NULL,
10.   PRIMARY KEY (`ID`),
11.   UNIQUE KEY `Name_UNIQUE` (`Name`),
12.   KEY `Host_School_idx` (`Host`),
```

```
  CONSTRAINT `Host_School` FOREIGN KEY (`Host`) REFERENCES `school` (`Name`) ON DELETE CASCADE ON UPDATE CASCADE
) ;

CREATE TABLE `members_of` (
  `Player` int NOT NULL,
  `Team` varchar(45) NOT NULL,
  `Role` enum('教练','队长','队员') DEFAULT NULL,
  PRIMARY KEY (`Player`,`Team`),
  KEY `members_of_team_ID_fk_idx` (`Team`),
  CONSTRAINT `members_of_players_ID_fk` FOREIGN KEY (`Player`) REFERENCES `players` (`ID`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `members_of_team_ID_fk` FOREIGN KEY (`Team`) REFERENCES `team` (`Name`) ON DELETE CASCADE ON UPDATE CASCADE
) ;

CREATE TABLE `participate_in` (
  `Team` varchar(45) NOT NULL,
  `Competition` varchar(45) NOT NULL,
  `Medal` enum('金奖','银奖','铜奖','优胜奖') DEFAULT NULL,
  `Place` int DEFAULT NULL,
  `Category` enum('正式','非正式') DEFAULT NULL,
  PRIMARY KEY (`Team`,`Competition`),
  KEY `participate-in_competition_Name_fk` (`Competition`),
  CONSTRAINT `participate-in_competition_Name_fk` FOREIGN KEY (`Competition`) REFERENCES `competition` (`Name`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `participate-in_team_Name_fk` FOREIGN KEY (`Team`) REFERENCES `team` (`Name`) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE `players` (
  `ID` int NOT NULL AUTO_INCREMENT,
  `Name` varchar(40) DEFAULT NULL,
  `Sex` enum('男','女') DEFAULT NULL,
  `E_mail` varchar(40) DEFAULT NULL,
  `School` varchar(40) DEFAULT NULL,
  `College` varchar(40) DEFAULT NULL,
  `Grade` enum('大一','大二','大三','大四','大五','研一','研二','研三','教师') DEFAULT NULL,
  PRIMARY KEY (`ID`),
  UNIQUE KEY `players_E-mail_uindex` (`E_mail`),
```

```
  KEY `players_school_Name_fk` (`School`),
  CONSTRAINT `players_school_Name_fk` FOREIGN KEY (`School`) REFERENCES `school` (`Name`) ON DELETE CASCADE ON UPDATE CASCADE
) ;

CREATE TABLE `school` (
  `ID` int NOT NULL AUTO_INCREMENT,
  `Name` varchar(45) DEFAULT NULL,
  `Nation` varchar(45) DEFAULT NULL,
  `Province` varchar(45) DEFAULT NULL,
  `City` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`ID`),
  UNIQUE KEY `Name_UNIQUE` (`Name`)
) ;

CREATE TABLE `team` (
  `ID` int NOT NULL AUTO_INCREMENT,
  `Name` varchar(45) DEFAULT NULL,
  `School` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`ID`),
  UNIQUE KEY `Name_UNIQUE` (`Name`),
  KEY `From_School_idx` (`School`),
  CONSTRAINT `From_School` FOREIGN KEY (`School`) REFERENCES `school` (`Name`) ON DELETE CASCADE ON UPDATE CASCADE
) ;
```

3.5 数据库应用设计

在本次实验中，我将数据库管理同用户界面隔离，这样便于后续数据库逻辑更改后应用的更新。

在开发应用的过程中，我利用 PyMySQL 库来建立应用和 MySQL 数据库的连接，使用 PyQt5 来实现图形化界面。我将所有负责数据管理的类放置在 data_manager 目录下，其中 MySQLConnect 类负责管理同 MySQL 数据库的连接，对一些重复率高的 MySQL 语句设置模版（如创建表、插入和查询语句）。如对于插入数据，我使用格式化字符串来实现。

```
cursor.execute("INSERT INTO {} ({}) VALUES ({});".format(table, keys, values))
```

而 CompetitionManager，MembersOfManager，ParticipateInManager，PlayerManager，SchoolManager，TeamManager 分别管理了比赛、人员队伍所属关系、参赛信息关系、人员、学校、队伍这几张表，他们通过 MySQLConnect 类来连接连接数据库，之后通过 MySQLConnect 类提供的 API 来更新和查询自己负责的数据表，为前端程序提供数据的查询和更新。

同 GUI 界面相关的配置文件和界面配置则存放在 ui 目录下。它们负责将数据直观地展示在用户面前，同时为用户提供相应的输入和输出方式，让用户能够通过一个简洁、友好的界面来对数据进行管理。

最后由 App 类负责将图形化界面同数据管理联系起来，将相关的按钮和函数关联起来，实现让用户进行数据管理。

实验题目	数据库设计与应用开发			实验日期	2022/5/1
班级	1903102	学号	1190200122	姓名	袁野

3.6 实现结果

首先我将整个系统根据操作的表的不同将其分为比赛、队伍、选手、学校四个页面，点击上方对应的按钮，并按照窗口的提示信息输入对应信息即可，如果输入信息不符合规范，系统会自动弹出错误提示并终止输入。

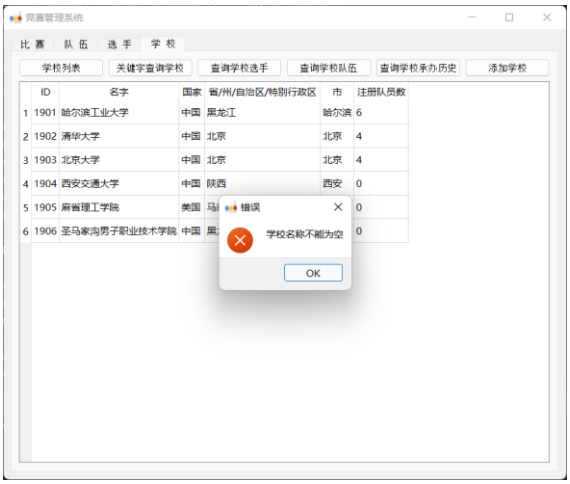

图 2 出错提示

学校页面支持的功能有：

- 学校列表：列出表中所有的学校以及包含注册人员数
- 关键字查询学校：输入字符进行学校名字模糊搜索
- 查询学校选手：列出输入学校的所有人员
- 查询学校队伍：列出输入学校的所有队伍
- 查询学校承办历史：列出输入学校承办过的比赛
- 添加学校

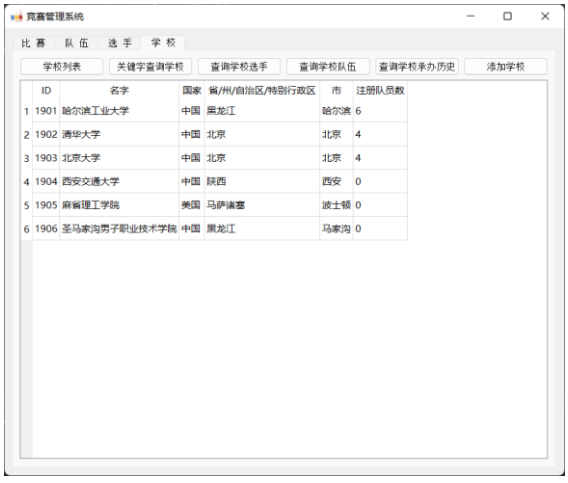

图 3 查询学校列表

实验题目	数据库设计与应用开发			实验日期	2022/5/1
班级	1903102	学号	1190200122	姓名	袁野

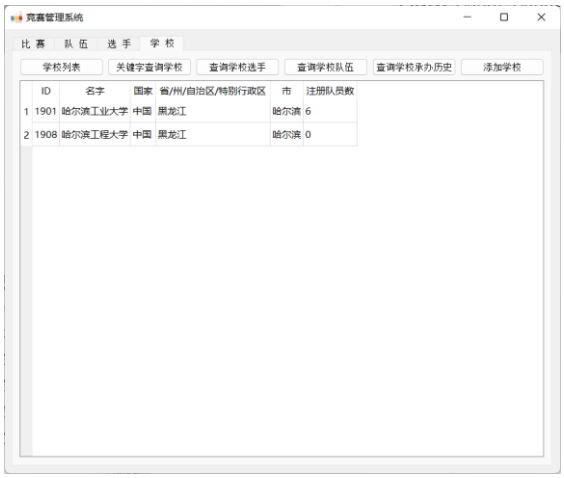

图 4 关键字搜索所有名字中带有“哈尔滨”的学校

选手页面支持的功能有：

- 选手列表：列出选手的详细信息
- 关键字查询选手：输入字符进行人员姓名、邮箱模糊搜索
- 精准查询选手：输入 ID 查询
- 查询选手荣誉：先根据选手查询所在的队伍集合，然后根据查询所有参赛记录里在这个队伍集合里的记录
- 添加选手：输入相关信息添加，其中邮箱部分会判断是否包含‘@’符号，不包含则反馈 Email 地址不正确

队伍页面支持的功能有：

- 队伍列表：列出所有队伍的信息
- 关键字查询队伍：输入字符进行队名模糊搜索
- 查询队伍成员：输入队名精确列出队伍的所有成员、教练的信息
- 查询队伍荣誉：根据输入队伍名输出参赛记录的详细信息
- 添加队伍：添加过程中会限制教练身份师是教师，队员、队长不是教师，且队员三个人不能重复

比赛页面支持的功能有：

- 比赛列表：列出所有比赛的信息
- 关键字查询比赛：输入字符进行比赛名模糊搜索
- 查询比赛结果：根据比赛名字列出所有该场比赛的参赛记录，并按照排名排序
- 添加比赛
- 添加比赛记录

四、实验结论（总结实验发现及结论）

在本次实验中，我学会了正确进行概念数据库设计，并将其用 ER 图展示出来，进一步进行逻辑数据库设计和物理数据库设计，同时将其设计的高效、规范、可读性强。此外掌握了数据库系统应用的开发，设计出了较为友好的简易的操作界面，提高了自己对 SQL 语言的熟悉程度。