数据库设计-运动比赛管理系统

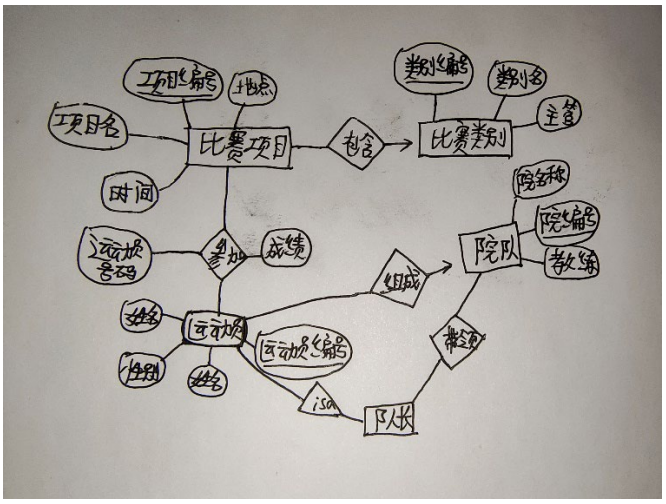
2212998　　胡博浩　　信息安全

一、详细需求描述

1. 比赛类别包括以下属性：
    - 类别编号 (CategoryID)：唯一标识一个比赛类别的编号。
    - 类别名称 (CategoryName)：描述比赛类别的名称。
    - 主管 (Supervisor)：负责管理该比赛类别的主管。
    - 比赛项目 (Events)：包含多个比赛项目，可通过类别编号进行识别。
2. 比赛项目包括以下属性：
    - 项目编号 (EventID)：唯一标识一个比赛项目的编号。
    - 项目名称 (EventName)：描述比赛项目的名称。
    - 比赛时间 (EventTime)：指定比赛项目的时间。
    - 比赛地点 (EventLocation)：指定比赛项目的地点。
    - 所属类别编号 (CategoryID)：指定该比赛项目所属的比赛类别。
3. 院队包括以下属性：
    - 院编号 (FacultyID)：唯一标识一个院队的编号。
    - 院名称 (FacultyName)：描述院队的名称。
    - 教练 (Coach)：负责指导院队的教练。
    - 队长 (Captain)：领导院队的队长。
    - 运动员 (Athletes)：由多名运动员组成，可通过院编号进行识别。
4. 运动员包括以下属性：
    - 运动员编号 (AthleteID)：唯一标识一个运动员的编号。
    - 姓名 (Name)：描述运动员的姓名。
    - 年龄 (Age)：指定运动员的年龄。
    - 性别 (Gender)：指定运动员的性别。
5. 每位运动员参加某个项目时会有以下属性：
    - 成绩 (Result)：记录运动员在某个项目中的成绩。
    - 号码 (BibNumber)：指定运动员在该项目中的参赛号码。
6. 每个项目可以有多名运动员参与，每名运动员可以参与多个项目，实现多对多的关系。

二、采用教材中介绍的方法实现如下设计

a)　绘制概念模型ER图



b) 将 ER 图转换为关系模式

以下是关系模式，包括主键（PK）和外键（FK）：

1. 比赛类别关系模式：
   - Category(CategoryID(PK), CategoryName, Supervisor)
2. 比赛项目关系模式：
   - Event(EventID(PK), EventName, EventTime, EventLocation, CategoryID(FK))
   - 外键：CategoryID 参考 Category 关系的 CategoryID 主键
3. 院队关系模式：
   - Faculty(FacultyID(PK), FacultyName, Coach, CaptainID(FK))
   - 外键：CaptainID 参考 Athlete 关系的 AthleteID 主键
4. 运动员关系模式：
   - Athlete(AthleteID(PK), Name, Age, Gender, FacultyID(FK))
   - 外键：FacultyID 参考 Faculty 关系的 FacultyID 主键
5. 参赛记录关系模式：
   - Participation(AthleteID(PK,FK), EventID(PK,FK), Result, BibNumber)
   - 外键：AthleteID 参考 Athlete 关系的 AthleteID 主键
   - 外键：EventID 参考 Event 关系的 EventID 主键

c) 用SQL语句创建上述关系模式

-- 创建比赛类别关系模式

```
CREATE TABLE Category (
    CategoryID INT PRIMARY KEY,
    CategoryName VARCHAR(255),
    Supervisor VARCHAR(255)
```

```
);

-- 创建比赛项目关系模式
CREATE TABLE Event (
    EventID INT PRIMARY KEY,
    EventName VARCHAR(255),
    EventTime DATETIME,
    EventLocation VARCHAR(255),
    CategoryID INT,
    FOREIGN KEY (CategoryID) REFERENCES
    Category(CategoryID)
);

-- 创建院队关系模式
CREATE TABLE Faculty (
    FacultyID INT PRIMARY KEY,
    FacultyName VARCHAR(255),
    Coach VARCHAR(255),
    CaptainID INT,
    FOREIGN KEY (CaptainID) REFERENCES
    Athlete(AthleteID)
);

-- 创建运动员关系模式
CREATE TABLE Athlete (
    AthleteID INT PRIMARY KEY,
    Name VARCHAR(255),
    Age INT,
    Gender VARCHAR(10),
    FacultyID INT,
    FOREIGN KEY (FacultyID) REFERENCES
    Faculty(FacultyID)
);

-- 创建参赛记录关系模式
CREATE TABLE Participation (
    AthleteID INT,
    EventID INT,
    Result DECIMAL(10,2),
    BibNumber INT,
    PRIMARY KEY (AthleteID, EventID),
    FOREIGN KEY (AthleteID) REFERENCES
    Athlete(AthleteID),
```

```
    FOREIGN KEY (EventID) REFERENCES
    Event(EventID)
);
```

d)该数据库模式上 5 个查询语句样例

1. 单表查询：查询所有运动员的姓名和年龄。

```
SELECT Name, Age FROM Athlete;
```

2. 多表连接查询：查询所有参加某个比赛项目的运动员及其所属院队名称。

```
SELECT Athlete.Name, Faculty.FacultyName
FROM Athlete
JOIN Faculty ON Athlete.FacultyID =
    Faculty.FacultyID
JOIN Participation ON Athlete.AthleteID =
    Participation.AthleteID
WHERE Participation.EventID = [YourEventID];
```

3. 多表嵌套查询：查询某个比赛项目中成绩排名前三名的运动员姓名和成绩。

```
SELECT Name, Result
FROM Athlete
JOIN Participation ON Athlete.AthleteID =
    Participation.AthleteID
WHERE Participation.EventID = [YourEventID]
AND Result IN (
    SELECT TOP 3 Result
    FROM Participation
    WHERE EventID = [YourEventID]
    ORDER BY Result DESC
);
```

4. EXISTS 查询：检查是否存在某个运动员参加了所有的比赛项目。

```
SELECT AthleteID, Name
FROM Athlete A
WHERE NOT EXISTS (
    SELECT *
    FROM Event E
    WHERE NOT EXISTS (
        SELECT *
        FROM Participation P
        WHERE P.AthleteID = A.AthleteID AND
    P.EventID = E.EventID
```

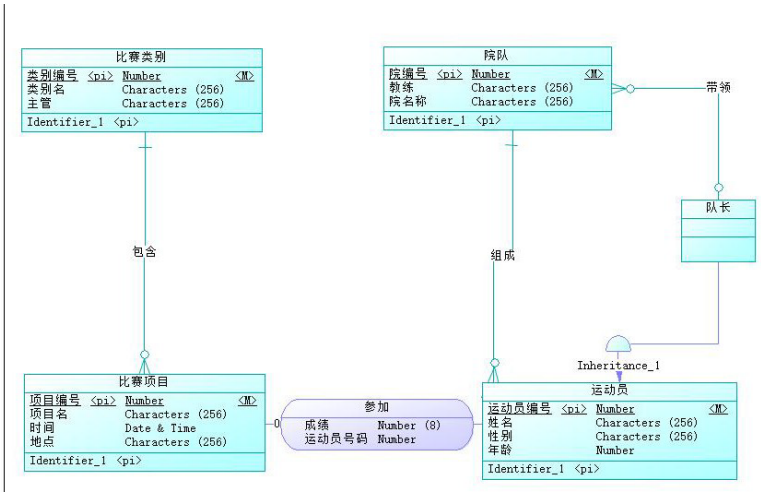
```
    )
);
```

5. 聚合操作查询：计算某个比赛项目中运动员的平均成绩。

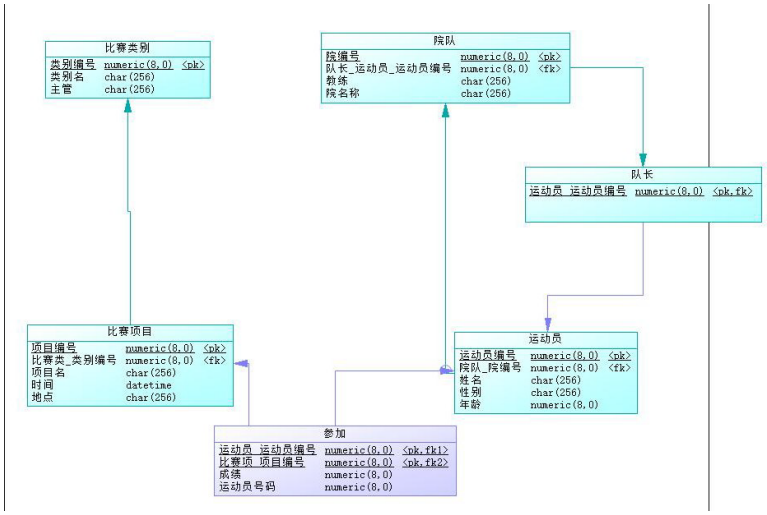
```
SELECT AVG(Result) AS AverageScore
FROM Participation
WHERE EventID = [YourEventID];
```

三、使用PowerDesigner实现如下设计

a) 概念模型ER图



b) 关系模型图



c) 生成创建数据库的SQL语句

```
drop table if exists 参加;
drop table if exists 比赛类别;
drop table if exists 比赛项目;
drop table if exists 运动员;
drop table if exists 队长;
drop table if exists 院队;
```

```
/*==============================================================*/
/* Table: 参加      */
/*==============================================================*/
create table 参加 (
运动员编号    numeric(8,0) not null,
项目编号            numeric(8,0) not null,
成绩   numeric(8,0),
运动员号码    numeric(8,0),
primary key (运动员编号, 项目编号)
);

/*==============================================================*/
/* Table: 比赛类别   */
/*==============================================================*/
create table 比赛类别 (

类别编号      numeric(8,0) not null,
类别名 char(256),
主管   char(256),
primary key (类别编号)
);

/*==============================================================*/
/* Table: 比赛项目   */
/*==============================================================*/
create table 比赛项目 (
项目编号      numeric(8,0) not null,
类别编号      numeric(8,0) not null,
项目名 char(256),
时间   datetime,
地点   char(256),
primary key (项目编号)
);

/*==============================================================*/
/* Table: 运动员     */
/*==============================================================*/
create table 运动员 (
运动员编号          numeric(8,0) not null,
院编号 numeric(8,0) not null,
姓名   char(256),
性别   char(256),
年龄   numeric(8,0),
primary key (运动员编号)
);

/*==============================================================*/
/* Table: 队长      */
/*==============================================================*/
create table 队长 (
运动员编号    numeric(8,0) not null,
primary key (运动员编号)
);

/*==============================================================*/
```

```
/* Table: 院队       */
/*==============================================================*/

create table 院队 (
院编号 numeric(8,0) not null,
运动员编号           numeric(8,0),
教练   char(256),
院名称 char(256),
primary key (院编号)
);

alter table 参加
       add constraint FK_参加 foreign key (运动员编号)
              references 运动员 (运动员编号);

alter table 参加
       add constraint FK_参加_2 foreign key (项目编号)
              references 比赛项目 (项目编号);

alter table 比赛项目
       add constraint FK_包含 foreign key (类别编号)
              references 比赛类别 (类别编号);

alter table 运动员
       add constraint FK_组成 foreign key (院编号)
              references 院队 (院编号);

alter table 队长
       add constraint FK_Inheritance_1 foreign key (运动员编号)
              references 运动员 (运动员编号);

alter table 院队
       add constraint FK_带领 foreign key (运动员编号)
              references 队长 (运动员编号);
```

四、分析比较上述两种方法

a)有巨大差异。

PowerDesigner引入了一个额外的表“队长”，来表示院队的队长。

优点：

1.扩展性：将队长信息单独提取到一个表中，使得在需要时可以更轻松地扩展队长的相关信息，而不会影响到运动员表的结构。

2.一致性：对于特定类型的用户，如队长，可以更加一致地管理他们的信息，与其他普通运动员区分开来。

缺点：

1.查询复杂性：引入了额外的表，查询涉及到队长信息时需要进行多表关联查询，增加了查询的复杂度。

2.维护复杂性：数据的维护需要更多的操作，例如更新队长信息时需要同步更新队长表和运动员表，增加了维护的难度。

3.数据完整性：需要额外的措施来保证队长表中的记录和运动员表中的记录保持一致性，避免出现数据不一致的情况。

这种设计的适用性取决于具体的需求和情况。如果对队长的信息需要进行更多的管理和扩展，并且能够接受查询和维护复杂性带来的影响，那么这种设计可能是合适的。然而，如果对查询和维护的复杂性有较高的要求，可能需要考虑其他更简单的设计方案。

b)PowerDesigner生成的SQL语句体现了模块化和简洁性的特点。

它将主键声明集中在每个表的末尾，并使用alter table语句来声明外键，这样做提高了SQL脚本的可读性和简洁性。此外，将外键约束放在alter table语句中也有助于将数据库的结构和关系清晰地表达出来，同时确保了数据库的完整性和一致性。这种设计能够有效地减少了冗余和复杂性，使得SQL脚本更易于理解和维护。

总的来说，Powerdesigner的附加语句排除了任何有可能干预数据库之实现的因素，这既是其出现的缘由，也是其作用。