足球联赛管理系统

**课程名**：数据库概论

**院系**：数学系

**年级**：2020级

**小组成员**：201840043 王一鸣

201840045 王佳鹏

**指导老师**：黄达明

**完成时间**：2021.12

目录：

一、组员分工

二、系统概述

（一）、设计初衷

（二）、系统功能

三、系统设计

（一）数据库设计

1.数据库建立

2.关系模型

3.表的设计

4.数据完整性

（二）功能设计与实现

1、按照不同关键字查询比赛的信息（包括球队、球员、日期）

2、查询当前联赛排名情况

3、按照队伍的名字查询某个队伍的基本情况

4、按照队伍的名字查询该队伍的现役球员

5、查询某个球员的信息

6、查询球员进球数排名

7、利用触发器自动更新数据

四、数据库恢复使用步骤

五、数据库和功能设计及测试中遇到的问题

一、组员分工

1.王一鸣：

构思数据库结构；建立触发器和存储过程；完善改正数据库功能；完成演示ppt；补充撰写设计报告。

2.王佳鹏：

提供设计创意；设计数据库的基本结构以及表格的框架；主要撰写设计报告；演示数据库的运行。

二、系统概述

（一）设计初衷

一个足球联赛中，球队数量依实际情况而定。正规的国家级联赛球队数大抵在18到22之间（如五大联赛、中超等）。每一年的联赛中两支不同球队共交手两次，其中主客场各一次。在每次比赛后各队按结果进行积分排名，赢一场得3分，平一场得1分，输一场不得分。每一轮每一支球队需进行一场比赛，若联赛总共有20支球队，则共需38轮所有球队互相交手两次。赛程结束后，球队按照积分高低决定冠亚军和升降级。（若积分相同则各联赛有相应规则、在下面不作考虑）。

身为各种球迷或者比赛的观众，我们常常会需要查询我们喜欢的球员、球队或者近期的比赛。而这些功能一般而言离不开数据库的建立。

此种类型的数据库虽然是生活中最常见的类型，我们却很少完整地探究这样一个看似简单的数据库是如何建立起来以及产生实际作用的。于是我们选择了足球联赛的管理数据库作为数据库大作业的主题。

为了模拟联赛数据库的管理，我们建立了三个表、八个存储过程、两个触发器以及若干测试用数据。主要涉及了球队、球员和比赛。一方面希望模拟用户使用数据库，给用户带来更加舒适的体验，另一方面从数据库管理员的角度，希望能够建立一个相对完整的数据库并具备一定的严谨性。

另外，数据库的建立也是对本学期所学数据库相关知识的一个应用。设计的过程中希望能深入地体会运用理论知识解决实际问题的方法。

（二）系统功能

1、按照不同关键字查询比赛的信息（包括球队、球员、日期）

2、查询当前联赛排名情况

3、按照队伍的名字查询某个队伍的基本情况

4、按照队伍的名字查询该队伍的现役球员

5、查询某个球员的信息

6、查询球员得分排名

7、检验插入的比赛信息是否有误

8、在正确插入一条比赛信息后更新相应的数据

三、系统设计

（一）数据库设计

1.数据库建立

主文件逻辑名：league

主文件物理名：d:\database\league.mdf

初始规模：100MB

最大规模：无限

增长幅度：10%

日志文件逻辑名：league\_log

日志文件物理名：d:\database\league\_log.ldf

初始规模：5MB

最大规模：100MB

增长幅度：5%

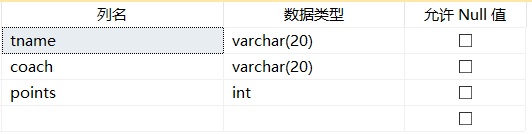
2.关系模型 ： 球队与球员的关系是典型的一对多关系，但就现实而言，我们区分球员的依据是他所属的球队和他的球衣号码。故我们这里把球队与球员的关系设计成两张表，将球队名称和球员球衣号码作为球员表的主键使用。比赛与球队和球员也相应地分别构成一对多关系，因此也把球队和球员表的主键分别放入比赛表中作为比赛表主键的一部分。

3.表的设计：

按照关系模型，共设计三张表

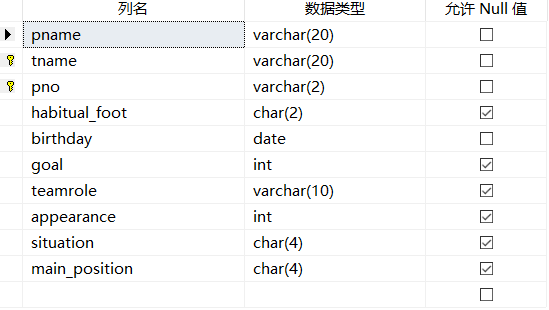
team表

列名依次为队伍名称（唯一标识），教练，当前积分



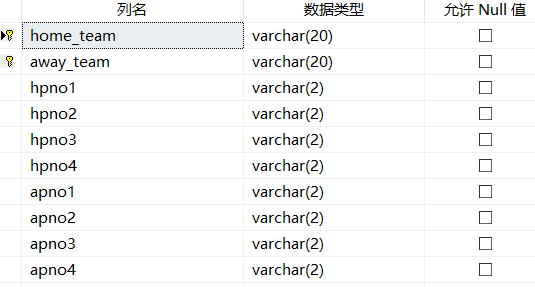
player表

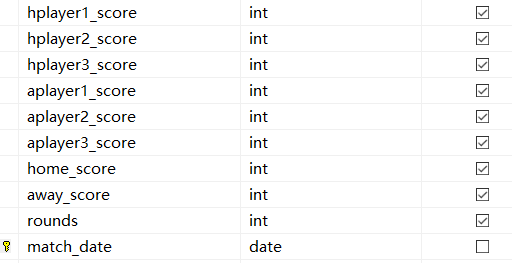
列名依次为球员姓名、所属队伍名称、球员球衣号码、惯用脚、生日、进球数、队伍角色（即是否为队长）、健康状况、主打位置



match表

列名依次为主队、客队、主客队各四位上场球员的球衣号码、主客队各三位球员的进球数（这里我们默认守门员不会得分，但是实际情况是允许的，不过对功能实现没有大的影响）、主队得分、客队得分、比赛轮次、比赛日期





注：关系模型在比赛对象中省略的具体主客场各四位球员名和这些球员各自的进球数在表中有所体现。

4.数据完整性：

（1）、约束

对球员的惯用脚添加约束，只能为“左”或“右”

对球员在队伍中的角色添加约束，只能为“队长”或“普通球员”

对球员当前的身体状况添加约束，只能为“健康”或“伤病”

对球员在赛场上主打的位置添加约束，为“前场，中场，后场，门将”四者之一

（2）、默认值

由于插入比赛信息时有可能比赛还没有进行，故此时参赛球员的得分都默认为0，而把两个队伍的得分设置为null

另外每个球队积分的默认值、球员的进球数和出场次数也默认为0

（3）触发器设置：

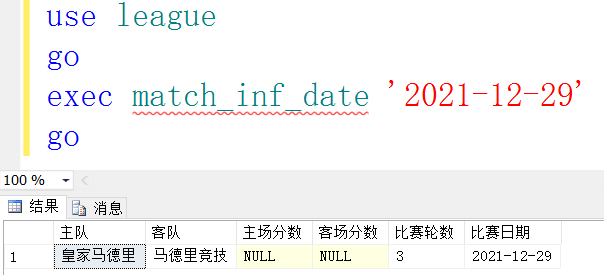
考虑联赛进行的情况，设置如下触发器。

首先在插入新的比赛信息的时候自动检验数据的正确性（比赛上场的球员是否与队伍一致），如果出现问题则会提示并回滚插入操作。

每插入一条赛程信息，球员表中的进球数、出场数均随比赛情况修改（助攻数等其他数据可以类似设置）。球队表中积分也随之修改，进而影响积分排名表。

(二)、功能设计与实现

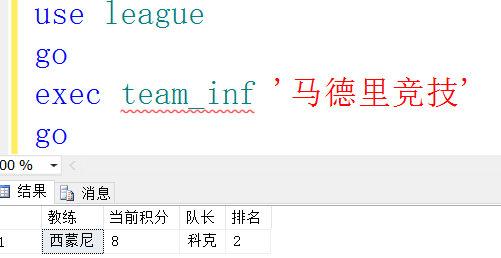
1、首先是查询比赛信息的功能。考虑到实际查询比赛信息时，我们可能会从喜欢的球员、球队，或是自己有时间观赛的日期三种不同的角度去查，因此创建了三个存储过程尽量满足实际需求。



球员



球队

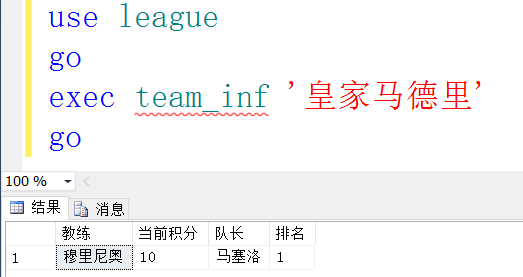


2、查询当前联赛排名情况

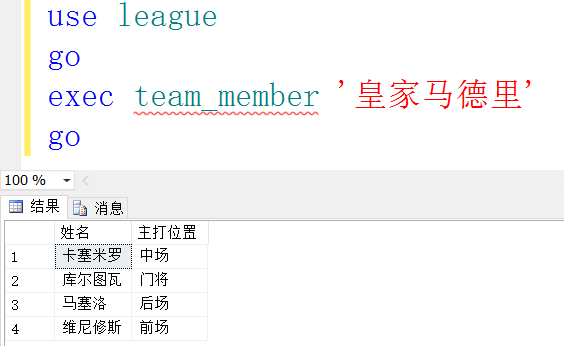
直接调动存储过程team\_rank就会显示当前联赛各球队的积分和排名情况



1. 按照队伍的名字查询某个队伍的基本情况



1. 按照队伍的名字查询该队伍的现役球员



5、查询某个球员的信息

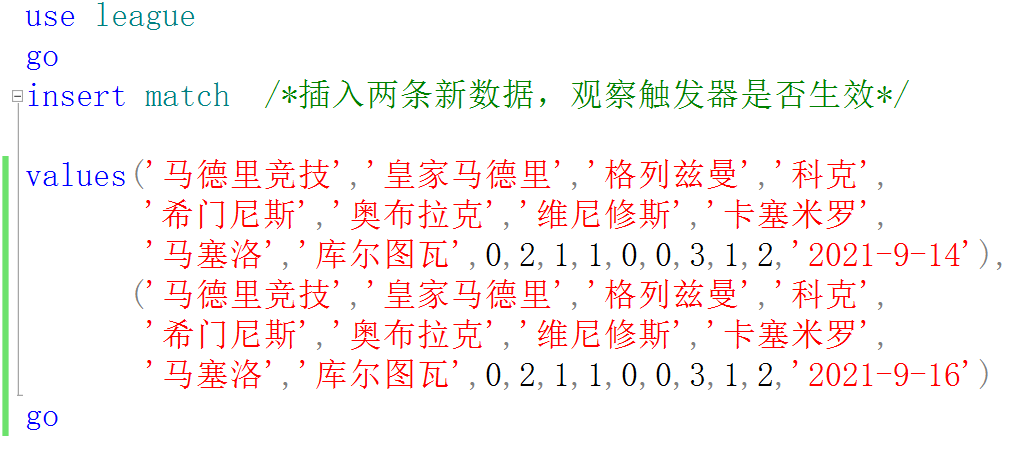


6.查询球员进球数排名



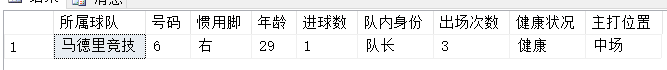
1. 利用触发器自动更新数据

检验触发器



更新前：





更新后：





四、备份与恢复

备份：将数据库league备份到磁盘文件d:\ sqlbackup\test\_backup.bak上，备份设备为物理设备

恢复：从磁盘文件d:\ sqlbackup\test\_backup.bak直接恢复数据库league

五、数据库和功能设计及测试中遇到的问题

1、在设计after insert触发器的时候，最初默认每次只插入一条数据，因此用select\*提取inserted表的数据并使用，但后来发现这对一次插入多条数据是无效的，因此采用了游标的方式去设计触发器。

2、设计表的时候出现了很多数据类型、约束、名称等的问题，后面通过反复调整才使得表相对合理。

3、在建表的时候，我们没有将队伍在联赛中的排名作为队伍的属性放在表中，因为这样的设计方式会需要建立一个相对复杂的触发器。但是这样带来的问题是需要在查询中将排名计算出来，而row\_number()等函数是对结果集的排名，并不满足我们的要求，故只能通过嵌套查询解决该问题。