Database Report - Rugby Union League

Description:

I designed a database model for a rugby league. The league would consist of five teams from different countries. Each team would play four games, playing each league member once. In the interest of fairness, the number of home and away games would be evenly split. My database consists of all the relevant information on my teams, players, stadia, fixtures, referees and results.

My "Stadium" entity simply would give the relevant information about a stadium. Because there is not a huge amount of stadia in the world, I assumed that the stadium name alone would be unique, thus I could make it the primary key. I also assumed that not every stadium has a specific team associated with it. A good example of this is the Aviva stadium, which has no particular club associated with it; however, it is used for fixtures that expect a larger attendance.

My "Team" table contains information about the teams name, location, sponsor, captain and mascot. It is assumed that the team name is unique, as no professional rugby teams have the same name. Thus, team name makes a suitable primary key. I gave each team a home stadium as a required attribute. This is because for any team to play in the professional era of rugby they need a home stadium. This stadium was a foreign key to the stadium entity. My captain attribute initially has no point of reference; however once I created the Player entity I could alter the table to make captain a foreign key to player.

My "Player" table takes in the relevant information about a player, i.e. age, name, id, salary etc. The primary key of this table is player id, a unique value for each player. This table has a foreign key to team, where it references which team the player plays for. This value can be NULL, as a player may be unsigned. I added another table, position, because one player can play in multiple positions. My position table simply records what position each player plays. Due to the fact that players can play in multiple positions, the table has a composite primary key consisting of player id and position.

My "Referee" table records the id, name, nationality, age and home club of each referee, with the id being the primary key.

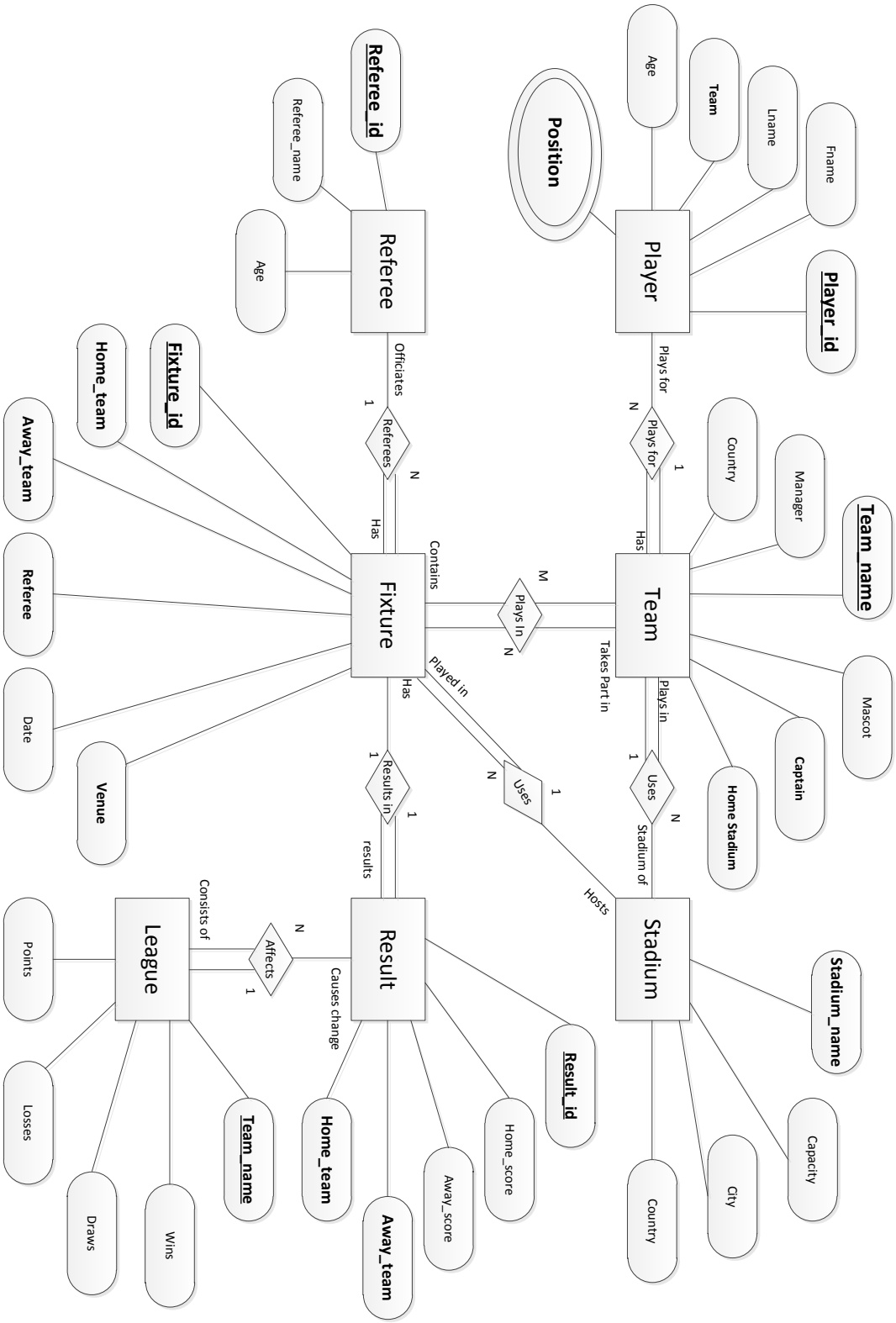
My "Fixture" entity records the home team, away team, referee and venue for each game. The fixture – team relationship is unusual, because unlike standard N:M relationships the relationship did not require a separate table. This is because the M value is a constant "2", one home team and one away team. A constraint in the creation of this table prevents the home team and away team being the same team. The fixture id is the primary key of this entity.

My "Results" entity takes the result of a given game, the important stats from that game and gives each game a unique result id. The result id is the foreign key for this table, and the fixture id is a foreign key to my fixture table.

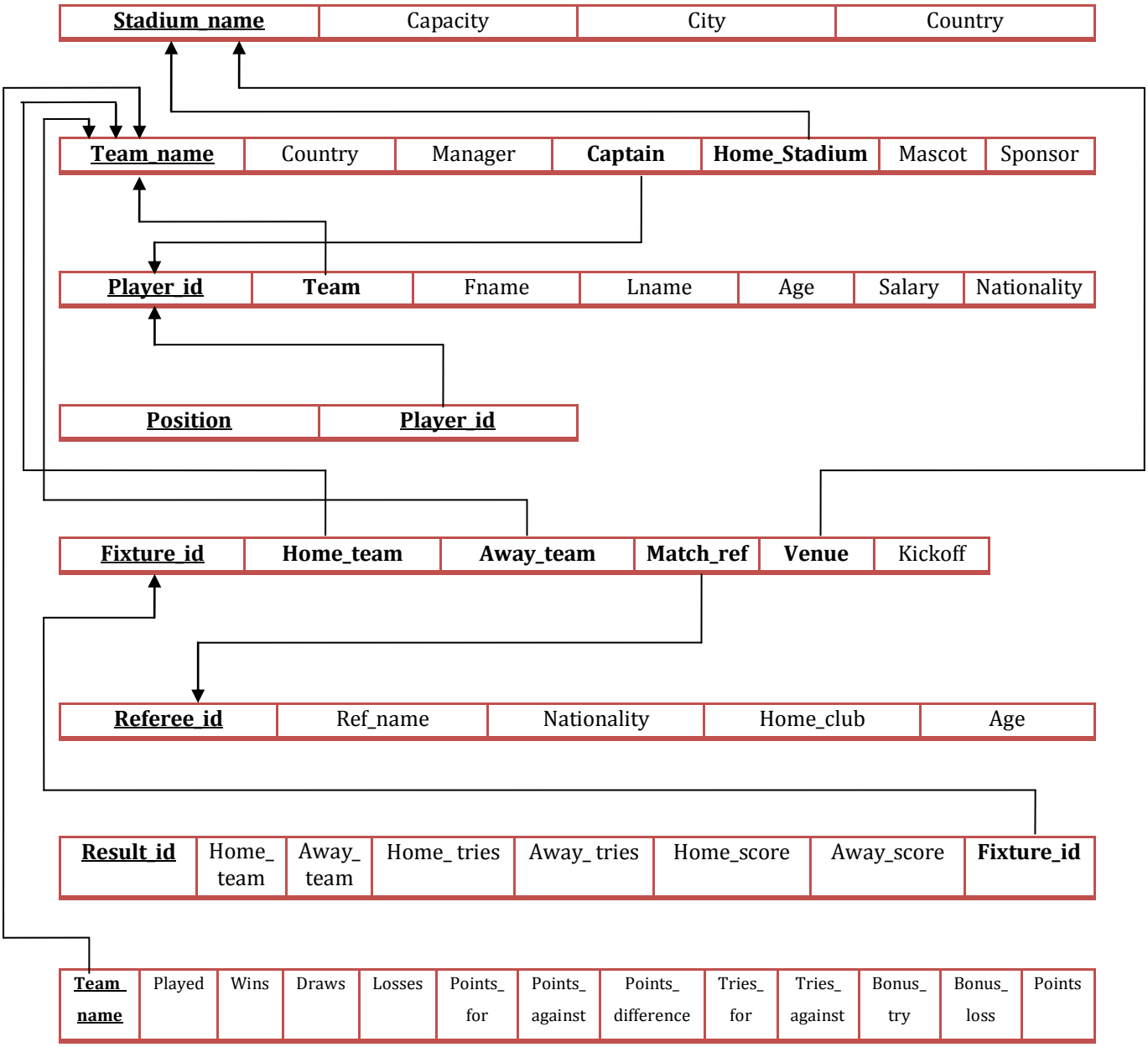
Finally I have a "League" table, which takes in the relevant values from the result table, and uses them to populate the overall league table.

Entity Relationship Diagram:

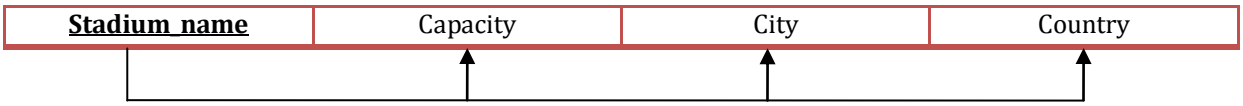
Shown below is the entity relationship I drew up for my database. NOTE: I have left out some excess attributes, as they do not add anything to the diagram (such as player salary) and they make my diagram look too cluttered.

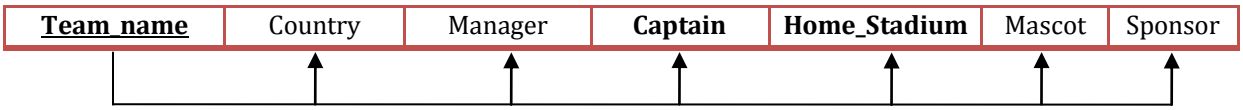


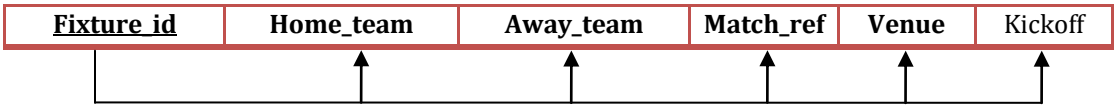
Relational Schema Diagram:

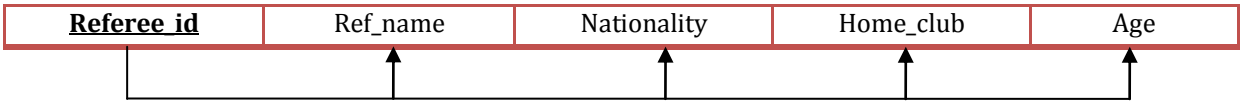


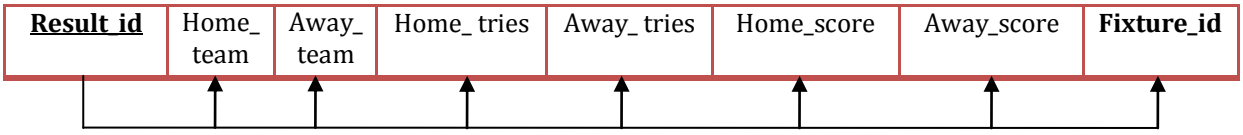
Functional Dependency:

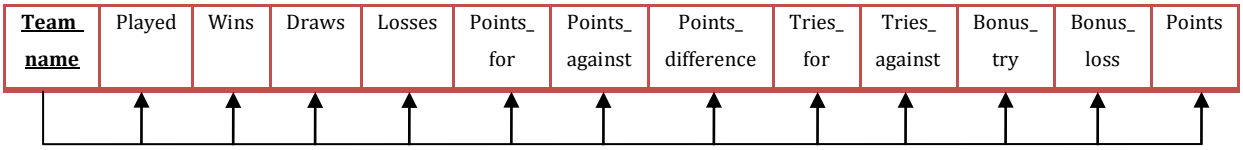












Normalisation:

The tables I used were almost entirely normalised before I drew up my functional dependencies. I had to add a new table, "Position" to avoid redundancy in my "Player". This new table had a composite primary key of "Player_id" and "Position". This allowed my database to be First Normal Form compliant, as now each attribute was atomic.

The second issue I noted earlier, wherein my Fixture – Team relation did not need its own table, as the number of teams per game was fixed. This meant that my database was not just Second Normal Form compliant, but also Third Normal Form Compliant, as every attribute in a table relied on the whole Primary Key and nothing but the Primary Key.

Explicit Constraints:

My primary keys (bold & underlined) and foreign keys (bold) are explicit constraints. Each Primary Key is unique and NOT NULL, thus avoiding integrity constraint violations. Secondly, my Foreign keys are mostly defined as NOT NULL to maintain referential integrity, however exceptions occur in the "Team" entity at "Captain", and the "Player" entity at the attribute "Team". These are because:

- When a new team is added to my tournament, it does not necessarily have a captain initially.
- A player may not currently be signed to a team. Players could sometimes be out of contract for a season

Semantic Constraints:

My tables had a number of semantic constraints applied to them. The most common constraint was ensuring that "id" tags were within their respective ranges. An example of this can be seen in my "Player" table, where "Player_id" must have a value between 100 and 999:

```
Constraint pid1_check Check (Player_id > 99),
Constraint pid2_check Check (Player_id < 999),
```

Another example of a semantic constraint comes from my two age checks. I felt that to be a professional rugby player a minimum age must be set. I decided that there could be no rugby players younger than 16, as shown in the code below:

```
Constraint p_age_check Check (Age > 16),
```

Similarly, my referees had to be at least 18 years of age. Another important constraint I set was to ensure that in any given fixture the home team and away team must be separate teams. This was to avoid human error in inputting data.

Some final constraints I could have placed include limiting the position inputs in "Position":

```
Constraint check_position Check (Position IN ('Prop', 'Hooker')),
```

However I felt this constraint would have been too long, and there are too many variable names for positions (i.e. outhalf, flyhalf and number 10 are all the same position and valid references to that position).

Another constraint I could have added was that referees could not officiate over matches from their home team, however in professional rugby there are not enough referees to achieve this goal at a club level, so I ignored this constraint, as does the real world. If this were an international tournament (i.e. The Rugby World Cup) then this constraint would be enforced, however this is merely a domestic league, so no constraint is required.

Finally, I made a number of my attributes NOT NULL, because a lot of the attributes are required to run a successful rugby league (such as match times or players age).

Views:

My database had two views automatically added, shown in Appendix E below. My first view, "Simple_League" simply takes some of the attributes from the "League" entity to show a condensed league table that is much easier to read. This view simply uses the SELECT statement to put the relevant information in the table. This view also uses the ORDER BY statement to order the teams from the team with the most points to the team with the least points.

My second view "Leinster_Fixtures" created a view that joined the Fixture information of Leinster, any final scores from games played by Leinster, and Leinster's current manager. To do this my table used the SELECT statement to obtain the relevant information, and used the WHERE statement to decide which information was relevant. As shown below, this table searches for any result in which Leinster was involved, and joins the information into one table using a table Join:

```
WHERE (Team.Team_name = Fixture.Home_Team OR Team.Team_name = Fixture.Away_Team)
    AND Team.Team_Name = 'Leinster' AND Fixture.Fixture_id = Results.Fixture_id;
```

This WHERE statement clearly uses a Join to connect the Fixture table and the Team table, and another Join to connect the Fixture table and the Result table.

Trigger Statement:

My trigger statement was quite a bit larger than the standard trigger statement; however there is good reason for this. My trigger is called every time a result is added. My trigger is designed to update for both the home team and the away team:

- Games Played.
- Wins, Draws and Losses.
- Points for, Points against and the overall Points Difference.
- Tries for and Tries against.
- Bonus points for scoring four or more tries in a game.
- Bonus points for losing by seven points or less.
- The overall points.

To achieve this task, my trigger has to first identify the home team. It then takes the relevant tuple from the "League" table, and performs the desired operation on it. This process is done for each column to be updated, and then the whole process is repeated for the away team's tuple in the "League" table. An example of this is shown below:

```
UPDATE League SET League.Wins = League.Wins + 1
WHERE (:new.Home_score > :new.Away_score AND League.Team_name = :new.Home_team);
```

This command updates the number of wins for the home team in the result if the team did in fact beat the opposition. If the team failed to beat the competition the prerequisites of the WHERE statement will not be met, and this UPDATE statement will be ignored.

Security:

The security of my database is an interesting issue, due to the amount of people involved in running a competition. For this database I would use the Database Management System (DBMS). This system operates by granting rights to specific users, allowing them to read and write information based on their privileges. The privileges would be bestowed upon users by a Database Administrator (DBA) who would be assigned by the tournament organiser. The DBA's powers in this instance would include:

- Creating user accounts
- Granting Privileges
- Revoking Privileges

In my database the DBA would rarely need to use his privileges, as most of the staff would remain constant throughout the tournament. Some users that may be required rights would be:

- A representative from each club, who could update their information and their players information (however the tournament organiser may also organise this).
- A representative from the referees association, who could update the Referee table.
- A member of the Scheduling committee, who would insert the Fixtures for the coming season into the table.
- An assistant who would insert the relevant information from the result of each game.

Depending on the size of the tournament, some of these may not be required, or additional users could be required to maintain the database. Combined with this, each club would have its own database which would accurately track their player's private information, such as medical information or social security numbers.

The users in my database would be granted mostly Read Privileges and Modification Privileges. This is because my database should not require new tables to be created, or tables to be deleted. An example of privileges I would give to the referee's representative is shown below:

GRANT INSERT, UPDATE, DELETE ON Referee TO Referee_rep WITH GRANT OPTION

This would allow the representative of the referees association to insert new referees into the referee table, or alter or delete tuples from the table. This command would also allow the representative of the referees association to pass on the privilege to other people. I would allow this because I can trust that the referees association to keep the privilege in safe hands so that the table will not face malicious corruption of its information. If I felt they were abusing their position I could remove their rights by:

REVOKE INSERT, UPDATE, DELETE ON Referee TO Referee_rep

Conveniently, this would also revoke the privileges from anybody the Referee_rep passed on the privileges to, so that once again only the DBA can alter the table.

Appendix A: Create Tables

```
Create Table Stadium
(
Stadium_name Varchar(20) NOT NULL,
Capacity INTEGER NOT NULL,
City Varchar(20) NOT NULL,
Country Varchar(20) NOT NULL,
Primary Key (Stadium_name)
);

Create Table Team
(
Team_name Varchar(10) NOT NULL,
Country Varchar(20) NOT NULL,
Manager Varchar(20) NOT NULL,
Captain INTEGER,
Home_Stadium Varchar(20) NOT NULL,
Mascot Varchar(20),
Sponsor Varchar(20),
Primary Key (Team_name),
Foreign Key (Home_Stadium) References Stadium (Stadium_name)
);

Create Table Player
(
Player_id INTEGER NOT NULL,
Team Varchar(10),
Fname Varchar(20) NOT NULL,
Lname Varchar(20) NOT NULL,
Age INTEGER NOT NULL,
Salary INTEGER,
Nationality Varchar(20) NOT NULL,
Primary Key (Player_id),
Constraint p_age_check Check (Age > 16),
Constraint pid1_check Check (Player_id > 99),
Constraint pid2_check Check (Player_id < 999),
Foreign Key (Team) References Team(Team_name)
);

Create Table Position
(
Player_id INTEGER NOT NULL,
Position Varchar(20) NOT NULL,
Primary Key (Player_id, Position),
Foreign Key (Player_id) References Player (Player_id)
);
```

```
Create Table Referee
(
Ref_id INTEGER NOT NULL,
Ref_name Varchar(20) NOT NULL,
Nationality Varchar(20) NOT NULL,
Home_club Varchar(10),
Age INTEGER NOT NULL,
Primary Key (Ref_id),
Constraint age_check Check (Age > 18),
Constraint id1_check Check (Ref_id > 9),
Constraint id2_check Check (Ref_id < 99)
);

Create Table Fixture
(
Fixture_id INTEGER NOT NULL,
Home_team Varchar(10) NOT NULL,
Away_team Varchar(10) NOT NULL,
match_ref INTEGER NOT NULL,
Venue Varchar(20) NOT NULL,
Kickoff DATE NOT NULL,
Primary Key (Fixture_id),
Constraint ref_check Check (Home_team != Away_team),
Constraint mid1_check Check (Fixture_id > 1000),
Constraint mid2_check Check (Fixture_id < 9999),
Foreign Key (Home_team) References Team (Team_name),
Foreign Key (Away_team) References Team (Team_Name),
Foreign Key (Venue) References Stadium (Stadium_name),
Foreign Key (match_ref) References Referee(Ref_id)
);

Create Table Results
(
Result_id INTEGER NOT NULL,
Home_team Varchar(10) NOT NULL,
Away_team Varchar(10) NOT NULL,
Home_score INTEGER,
Away_score INTEGER,
Home_tries INTEGER,
Away_tries INTEGER,
Fixture_id INTEGER NOT NULL,
Primary Key (Result_id),
Foreign Key (Fixture_id) References Fixture(Fixture_id)
);
```

```
Create Table League
(
Team_name Varchar(10) NOT NULL,
Played INTEGER,
Wins INTEGER,
Draws INTEGER,
Losses INTEGER,
Points_for INTEGER,
Points_against INTEGER,
Points_difference INTEGER,
Tries_for INTEGER,
Tries_against INTEGER,
Bonus_try INTEGER,
Bonus_loss INTEGER,
Points INTEGER,
Primary Key (Team_name),
Foreign Key (Team_name) References Team(Team_name)
);
```

Appendix B: Trigger

```
CREATE TRIGGER Match_complete
AFTER INSERT ON Results
FOR EACH ROW
BEGIN
UPDATE League SET League.Played = League.Played + 1
WHERE League.Team_name = :new.Home_team;
UPDATE League SET League.Points_for = League.Points_for + :new.Home_score
        WHERE (League.Team_name = :new.Home_team);
UPDATE League SET League.Points_against = League.Points_against + :new.Away_score
WHERE (League.Team_name = :new.Home_team);
UPDATE League SET League.Points_difference = League.Points_for - League.Points_against
WHERE (League.Team_name = :new.Home_team);

UPDATE League SET League.Tries_for = League.Tries_for + :new.Home_tries
WHERE (League.Team_name = :new.Home_team);
UPDATE League SET League.tries_for = League.Tries_against+ :new.Away_tries
WHERE (League.Team_name = :new.Home_team);

UPDATE League SET League.Wins = League.Wins + 1
WHERE (:new.Home_score > :new.Away_score AND League.Team_name = :new.Home_team);
UPDATE League SET League.Points = League.Points + 4
WHERE (:new.Home_score > :new.Away_score AND League.Team_name = :new.Home_team);

UPDATE League SET League.Draws = League.Draws + 1
WHERE (:new.Home_score = :new.Away_score AND League.Team_name = :new.Home_team);
UPDATE League SET League.Points = League.Points + 2
WHERE (:new.Home_score = :new.Away_score AND League.Team_name = :new.Home_team);

UPDATE League SET League.Losses = League.Losses + 1
WHERE (:new.Home_score < :new.Away_score AND League.Team_name = :new.Home_team);

UPDATE League SET League.Bonus_try = League.Bonus_try +1
WHERE (:new.Home_tries >3 AND League.Team_name = :new.Home_team);
UPDATE League SET League.Points = League.Points+1
WHERE (:new.Home_tries >3 AND League.Team_name = :new.Home_team);

UPDATE League SET League.Bonus_try = League.Bonus_loss +1
WHERE ((:new.Home_score < :new.Away_score) AND (:new.Away_Score- :new.Home_score) < 8 AND
League.Team_name = :new.Home_team);

UPDATE League SET League.Points = League.Points+1
WHERE ((:new.Home_score < :new.Away_score) AND (:new.Away_Score- :new.Home_score) < 8 AND
League.Team_name = :new.Home_team);
```

```
UPDATE League SET League.Played = League.Played + 1
WHERE League.Team_name = :new.Away_team;
UPDATE League SET League.Points_for = League.Points_for + :new.Away_score
        WHERE (League.Team_name = :new.Away_team);
UPDATE League SET League.Points_against = League.Points_against + :new.Home_score
WHERE (League.Team_name = :new.Away_team);
UPDATE League SET League.Points_difference = League.Points_for - League.Points_against
WHERE (League.Team_name = :new.Away_team);

UPDATE League SET League.Tries_for = League.Tries_for + :new.Away_tries
WHERE (League.Team_name = :new.Away_team);
UPDATE League SET League.tries_for = League.Tries_against+ :new.Home_tries
WHERE (League.Team_name = :new.Away_team);

UPDATE League SET League.Losses = League.Losses + 1
WHERE (:new.Home_score > :new.Away_score AND League.Team_name = :new.Away_team);

UPDATE League SET League.Draws = League.Draws + 1
WHERE (:new.Home_score = :new.Away_score AND League.Team_name = :new.Away_team);
UPDATE League SET League.Points = League.Points + 2
WHERE (:new.Home_score = :new.Away_score AND League.Team_name = :new.Away_team);

UPDATE League SET League.Wins = League.Wins + 1
WHERE (:new.Home_score < :new.Away_score AND League.Team_name = :new.Away_team);
 UPDATE League SET League.Points = League.Points + 4
WHERE (:new.Away_score > :new.Home_score AND League.Team_name = :new.Away_team);


UPDATE League SET League.Bonus_try = League.Bonus_try +1
WHERE (:new.Away_tries >3 AND League.Team_name = :new.Away_team);
UPDATE League SET League.Points = League.Points+1
WHERE (:new.Away_tries >3 AND League.Team_name = :new.Away_team);

UPDATE League SET League.Bonus_loss = League.Bonus_loss +1
WHERE ((:new.Away_score < :new.Home_score) AND (:new.Home_Score- :new.Away_score) < 8 AND
League.Team_name = :new.Away_team);

UPDATE League SET League.Points = League.Points+1
WHERE ((:new.Away_score < :new.Home_score) AND (:new.Home_Score- :new.Away_score) < 8 AND
League.Team_name = :new.Away_team);

END Match_complete;
/
```

Appendix C: Populating the Database

```
Insert into Stadium Values ('Landsdowne Road', 55000, 'Dublin', 'Ireland');
Insert into Stadium Values ('The RDS', 55000, 'Dublin', 'Ireland');
Insert into Stadium Values ('Ravenhill', 55000, 'Antrim', 'Ireland');
Insert into Stadium Values ('Parc Y Scarlets', 55000, 'LLanelli', 'Wales');
Insert into Stadium Values ('Stadio de Monigo', 55000, 'Veneto', 'Italy');
Insert into Stadium Values ('Murrayfield', 55000, 'Edinburgh', 'Scotland');

Insert into Team Values ('Leinster', 'Ireland', 'O Connor', 100, 'The RDS','Leo the Lion', 'BOI');
Insert into Team Values('Ulster', 'Ireland', 'Anscombe', 200, 'Ravenhill','masc1', 'Belfast Telegraph');
Insert into Team Values('Scarlets', 'Wales', 'Easterby', 300 , 'Parc Y Scarlets','masc2', 'Sponsor1');
Insert into Team Values('Treviso', 'Treviso', 'Smith', 400, 'Stadio de Monigo','masc3', 'Sponsor2');
Insert into Team Values('Edinburgh', 'Scotland', 'Solomons', 500, 'Murrayfield','masc4', 'Sponsor3');

Insert into Player Values (100, 'Leinster', 'Leo', 'Cullen', 35, 50000, 'Ireland');
Insert into Player Values (200, 'Ulster', 'Rory', 'Best', 20, 55000, 'Ireland');
Insert into Player Values (300, 'Scarlets', 'Capt', 'W', 30, 40000, 'Wales');
Insert into Player Values (400, 'Treviso', 'Capt', 'I', 29, 35000, 'Italy');
Insert into Player Values (500, 'Edinburgh', 'Capt', 'S', 19, 25000, 'Australia');

Insert into Position Values (100, 'Lock');
Insert into Position Values(200, 'Hooker');
Insert into Position Values(300, 'Wing');
Insert into Position Values(300, 'Full Back');
Insert into Position Values(400, 'Outhalf');
Insert into Position Values(400, 'Scrumhalf');
Insert into Position Values(400, 'Inside Center');
Insert into Position Values(500, 'Flanker');


Insert into Referee Values (11, 'Nigel Owens', 'Wales', 'Scarlets', 30);
Insert into Referee Values (12, 'Alan Roland', 'Ireland', 'Leinster', 40);
Insert into Referee Values (13, 'Roman Poitre', 'France', NULL, 35);
Insert into Referee Values (14, 'Whoever', 'Scotland', 'Edinburgh', 20);
Insert into Referee Values (15, 'Yer Man', 'South Africa', NULL, 25);
```

```
Insert into Fixture Values(1001,'Leinster','Treviso',11,'The RDS', TO_DATE ('2013-09-11 18:00','yyyy-mm-dd HH24:Mi'));
Insert into Fixture Values(1002,'Ulster','Edinburgh',13,'Ravenhill', TO_DATE ('2013-09-11 20:00','yyyy-mm-dd HH24:Mi'));
Insert into Fixture Values(1003,'Scarlets','Leinster',15,'Parc Y Scarlets', TO_DATE('2013-09-18 18:00','yyyy-mm-dd HH24:Mi'));
Insert into Fixture Values(1004,'Treviso','Edinburgh',12,'Stadio de Monigo', TO_DATE('2013-09-18 20:00','yyyy-mm-dd HH24:Mi'));
Insert into Fixture Values(1005,'Scarlets','Treviso',12,'Parc Y Scarlets', TO_DATE('2013-10-02 18:00','yyyy-mm-dd HH24:Mi'));
Insert into Fixture Values(1006,'Leinster','Ulster',14,'Landsdowne Road', TO_DATE('2013-10-02 20:00','yyyy-mm-dd HH24:Mi'));
Insert into Fixture Values(1007,'Edinburgh','Leinster',11,'Murrayfield', TO_DATE('2013-10-16 18:00','yyyy-mm-dd HH24:Mi'));
Insert into Fixture Values(1008,'Ulster','Scarlets',13,'Ravenhill', TO_DATE('2013-10-16 20:00','yyyy-mm-dd HH24:Mi'));
Insert into Fixture Values(1009,'Edinburgh','Scarlets',15,'Murrayfield', TO_DATE('2013-10-23 18:00','yyyy-mm-dd HH24:Mi'));
Insert into Fixture Values(1010,'Treviso','Ulster',14,'Stadio de Monigo', TO_DATE('2013-10-23 20:00','yyyy-mm-dd HH24:Mi'));

Insert into League Values ('Leinster', 0,0,0,0, 0,0,0, 0,0,0,0, 0);
Insert into League Values('Ulster', 0,0,0,0, 0,0,0, 0,0,0,0, 0);
Insert into League Values('Scarlets', 0,0,0,0, 0,0,0, 0,0,0,0, 0);
Insert into League Values('Treviso', 0,0,0,0, 0,0,0, 0,0,0,0, 0);
Insert into League Values('Edinburgh', 0,0,0,0, 0,0,0, 0,0,0,0, 0);

Insert into Results Values('10001', 'Leinster', 'Treviso', 50, 8, 5, 1, 1001);
Insert into Results Values('10002', 'Ulster', 'Edinburgh', 26, 22, 4, 3, 1002);
Insert into Results Values('10003', 'Scarlets', 'Leinster', 0, 7, 0, 1, 1003);
Insert into Results Values('10004', 'Treviso', 'Edinburgh', 15, 15, 2, 2, 1004);
Insert into Results Values('10005', 'Scarlets', 'Treviso', 25, 7, 3, 1, 1005);
Insert into Results Values('10006', 'Leinster', 'Ulster', 35, 28, 5, 4, 1006);
Insert into Results Values('10007', 'Edinburgh', 'Leinster', 12, 15, 2, 0, 1007);
Insert into Results Values('10008', 'Ulster', 'Scarlets', 32, 3, 4, 0, 1008);
Insert into Results Values('10009', 'Edinburgh', 'Scarlets', 30, 8, 5, 1, 1009);
Insert into Results Values('10010', 'Treviso', 'Ulster', 18, 20, 2, 4, 1010);
```

Appendix D: Alter

```
Alter table Team Add Foreign Key (Captain) References Player(Player_id);
```

Appendix E: Create View

```
Create View Simple_League AS
SELECT Team_Name,Wins,Draws,Losses,Points_difference,Points
FROM League ORDER BY Points DESC;

Create View Leinster_Fixtures(Match_id, Home_team, Away_team, Venue, Team_Manager, Home_score,
Away_Score) AS
SELECT Fixture.Fixture_id, Fixture.Home_Team, Fixture.Away_Team, Fixture.Venue, Team.Manager,
                Results.Home_score, Results.Away_score
FROM Team, Fixture, Results
WHERE (Team.Team_name = Fixture.Home_Team OR Team.Team_name = Fixture.Away_Team)
AND Team.Team_Name = 'Leinster' AND Fixture.Fixture_id = Results.Fixture_id;
```

Appendix F: Drop Tables/Views/Triggers

```
DROP TABLE Team CASCADE CONSTRAINTS;
DROP TABLE Stadium CASCADE CONSTRAINTS;
DROP TABLE Player CASCADE CONSTRAINTS;
DROP TABLE Position CASCADE CONSTRAINTS;
DROP TABLE Fixture CASCADE CONSTRAINTS;
DROP TRIGGER Match_complete;
DROP TABLE Results CASCADE CONSTRAINTS;
DROP TABLE Referee CASCADE CONSTRAINTS;
DROP TABLE League CASCADE CONSTRAINTS;
DROP VIEW Simple_League;
DROP VIEW Leinster_Fixtures;
```