

PES UNIVERSITY

100 feet Ring Road, BSK 3rd Stage, Bengaluru 560085

Department of Computer Science and Engineering

Jan – May 2020

UE18CS252

Database Management Systems

Project Report

English Premier League

Database Management

System

PES1201801765, Vignesh Veeresh Anweri

4th Sem Sec. A, Roll No. 50

|  |
| --- |
| **PROJECT SUMMARY**  The English Premier consists of 20 teams, each of which has a unique team name and ID. Each team belongs to a particular city. A city can have more than 1 team playing in the league . Teams play one another in a series of scheduled matches of 38 matches per team. Points are awareded based on wins, draws.  The team with maximum points at the end of the season is declared the winner. The league standings are updated after every insert of a match result. Triggers to restrict more than 20 teams, check attendace is not more than capacity of stadium and real time update of team statistics has been given. Information about manager of youngest player, total goals scored by a team, top goal scorer etc have been queried.  The system is capabale of updating information in real time based on matches played. It can be further used for other football leagues like the bundesliga, Seria A etc. |

[**Introduction**](#_gwqiqapyxr0x) **1**

[**Data Model**](#_p451stoz4u81) **2**

[**FD and Normalization**](#_wc0v2611q54c) **8**

[**DDL**](#_pn441yprtk2f) **16**

[**Triggers**](#_a9bswny9ffj) **21**

[**SQL Queries**](#_exvna2br67rz) **25**

[**Conclusion**](#_k6bbxckcf51j) **28**

**INTRODUCTION**

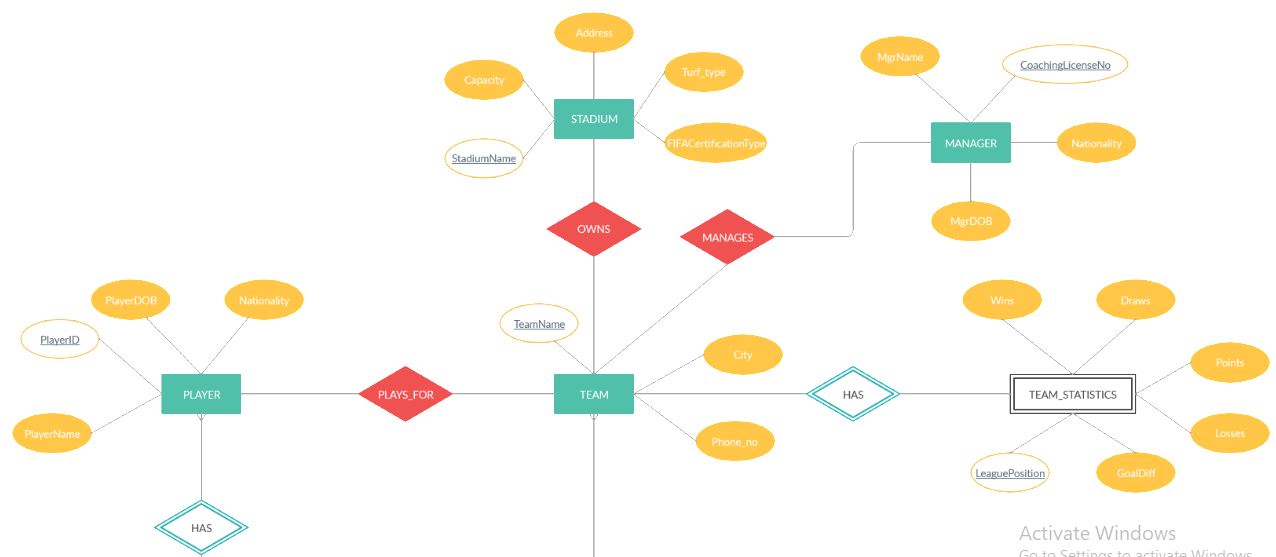
1. The English Premier consists of 20 teams, each of which has a unique team name and ID. Each team belongs to a particular city. A city can have more than 1 team playing in the league .
2. Each team consists of a number of players. No player belongs to more than one team. Each player is assigned a unique Player ID. The league also records each player’s name (ﬁrst name and last name), and date of birth. Each Player plays in a particular position, is of a particular nationality, has his own kit number etc.
3. Every players has his own statistics. The statistics are number of goals scored, red cards, yellow cards, number of suspensions etc.
4. Every team has a manager that manages it who has his details like the name, date of birth, nationality and coaching-License-Number. A manager can manage only one team at a time.
5. Teams play one another in a series of scheduled matches. Each match is given a unique Match ID number and is scheduled for a speciﬁc date and time(local time) during a given match-week.The league has a total of 38 matches. Each match is between two teams. All teams play a season schedule consisting of 38 matches. The league records the score of each match. In addition, the league records, for each match, the number of goals scored.
6. Every team has a unique name.
7. Each team is also assigned points based on wins(3 points), and draws(1 point), and losses(0 points). Based on points, the team is given a position in the league. The teams’ wins, draws and losses, goal difference is also recorded.
8. Each Match has a group of referees that officiates the match. Each Referee has his own, Fifa-Refree-License-number, DOB, and nationality.
9. Every match is played in a particular stadium. Each team has its own home-stadium. Every stadium is identified by its unique name. Every stadium has turf type, FIFA certification and a limited amount of capacity.
10. Each match is is broadcasted by a number of broadcasters based on region. They are identified by name and Broadcaster-id. Each region can have only one broadcaster.

Transactions of the system

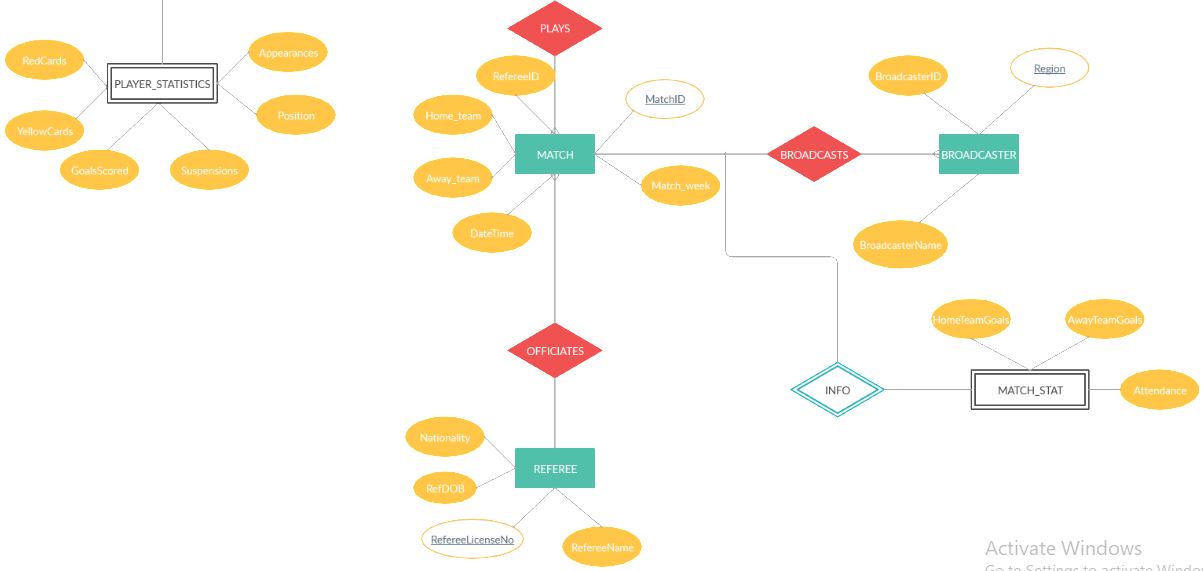
Everytime the match results are inserted, the team statistics , i.e the team’s wins, losses, draws, points, league position etc gets updated. The system does not let the number of teams playing exceed 20(throws an error incase).

The player statistics are also updated after every match. The user can find out the information about the league’s like the top scorer, youngst players, team at top of the league etc.

When a player joins or leaves a team during the course of the league his record gets added or deleted repectively.

**DATA MODEL**

ER Model



**PLAYER**

Primary key – PlayerID

Player ID is always unique for a given player.

Foreign key – PlayerTeam

The team a player plays for is referenced from the TEAM entity. A player plays for(relationship) a team.

Candidate key – PlayerID, (KitNo, PlayerTeam)

KitNo for a given player in a team is always unique.

**PLAYER\_STAT**

Primary Key – PlayerID

Player ID is always unique for a given player. PLAYER\_STAT is a weak entity, so PLAYER entity acts as an owner entity type. PLAYER has an identifying relationship with PLAYER\_STAT.

Foreign Key – PlayerID

As PLAYER\_STAT is a weak entity, it doesn’t have a primary key of its own. Primary key PlayerID is referenced as a foreign key from PLAYER entity.

Candidate key – PlayerID

**TEAM**

Primary Key – TeamName

A team name should always be unique. No two teams can have the same name. Hence team name can be used to identify a given team.

Foreign Key – Null

Candidate key – PhoneNo, TeamName

PhoneNo are always unique for a given person or organization.

**TEAM\_STAT**

Primary Key – TeamName

TeamName is always unique for a given team. TEAM\_STAT is a weak entity, so TEAM entity acts as an owner entity type. TEAM has an identifying relationship with TEAM\_STAT.

Foreign Key – TeamName

As TEAM\_STAT is a weak entity, it doesn’t have a primary key of its own. Primary key TeamName is referenced as a foreign key from TEAM entity.

Candidate key – TeamName, LeaguePosition

No two teams can have the same position in the league. Thus LeaguePosition is a unique.

**STADIUM**

Primary Key – StadiumName

The stadium owned by a given team must always have a unique name. Hence the stadium name can always be used to identify a stadium tuple.

Foreign Key – Team

Every team owns a stadium. STADIUM entity is related to TEAM entity through Team attribute which references TeamName of TEAM entity.

Candidate key – Team, StadiumName, Address

Address is always used to identtify a given location, hence must always be unique.

A team always own a one stadium only.

**MANAGER**

Primary Key – CoachingLicenseNo

Every coach is issued a unique coaching license. This can be used to uniqeuly identify a team manager.

Foreign Key – MgrTeam

A manager manages a team. TEAM and MANAGER entities are related by atrribute MgrTeam which references TeamName attribute from TEAM entity.

Candidate key – MgrTeam, CoachingLicenseNo

Every team can have only one manager and every manager can manage only one team. Hence MgrTeam is always unique and can be used to identify a tuple in the relation.

**MATCH**

Primary Key– MatchID

Every match played between two teams has a unique MatchID.

Foreign Key – RefereeID, HomeTeam, AwayTeam

A match is always officiated by a referee. The refereeID is referenced from RefereeLicenseNo attribute of entity REFEREE. This is how MATCH and REFEREE entities are related.

Candidate key – MatchID, (HomeTeam, MDateTime), (AwayTeam, HomeTeam)

On a given date, time and venue(HomeTeam determines which stadium the match is played in), only one match can take place. Hence (HomeTeam, MDateTime) is always unique.

Also 2 teams can face each other only twice, once home and the other away.

**MATCH\_STAT**

Primary Key – MatchID

MatchID is always unique for a given match. MATCH\_RESULT is a weak entity, so MATCH entity acts as an owner entity type. MATCH has an identifying relationship with MATCH\_RESULT.

Foreign Key – MatchID

As MATCH\_RESULT is a weak entity, it doesn’t have a primary key of its own. Primary key MatchID is referenced as a foreign key from MATCH entity.

Candidate Key - MatchID

**REFEREE**

Primary Key – RefereeLicenseNo

Each referee is issued a unique RefereeLicenseNo, which ca be used to identify a given referee.

Foreign Key – Null

Candidate key – RefereeLicenseNo

**BROADCASTER**

Primary Key – Region

Each region can have only one broadcaster. But since one broadcaster can broadcast in more than one region, we don’t take BroadcasterID as primary key. Hence we identify broadcaster tuple by Region

Foreign Key – Null

Candidate key – Region

**Relaional Database Schema**

PLAYER

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PlayerID | PlayerName | PlayerDOB | PlayerNationality | PlayerTeam | KitNo |

PLAYER\_STAT

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| PlayerID | PlayerPosition | Appearances | GoalsScored | YellowCards | RedCards | Suspensions |

TEAM

|  |  |  |
| --- | --- | --- |
| TeamName | City | PhoneNo |

TEAM\_STAT

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| TeamName | Wins | Draws | Losses | GoalDiff | Points | LeaguePosition |

STADIUM

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| StadiumName | Team | Address | Capacity | TurfType | FifaCertification |

MANAGER

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ManagerName | MgrTeam | CoachingLicenseNo | MgrDOB | MgrNationality |

MATCH

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MatchID | MatchWeek | HomeTeam | AwayTeam | MDateTime | RefereeID |

MATCH\_STAT

|  |  |  |  |
| --- | --- | --- | --- |
| MatchID | HomeTeamGoals | AwayTeamGoals | Attendance |

REFEREE

|  |  |  |  |
| --- | --- | --- | --- |
| RefereeName | RefereeLicenseNo | RefDOB | RefNationality |

BROADCASTER

|  |  |  |
| --- | --- | --- |
| BroadcasterID | BroadcasterName | Region |

**FDs** and **Normalisation**

**PLAYER**

**FDs**

PlayerID -----> PlayerName, PlayerDOB, PlayerNationality, PlayerTeam, kitNo

KitNo, PlayerTeam ------> PlayerName, PlayerDOB, PlayerNationality

**1NF**

Each attribute consists of atomic values and each value is single valued. Hence the given relation is in 1NF.

Suppose the player has dual citizenship, then the 1NF doesn’t hold good. But PlayerNationality here can only be the country the player plays for, which can only be a single value.

**2NF**

A single attribute primary key PlayerID can determine all prime and non-prime attributes, thus we don’t have any partial dependecies. Since the given relation is in 1NF and has no partial dependecies, the relation is also in 2NF.

Suppose the relation had no attribute PlayerID, and thus the primary key would be composite key (KitNo, PlayerTeam). Now if we had an attribute called PlayerManager(Manager of the player), the PlayerTeam can independently identify the manager of the player.

(KitNo, PlayerTeam) ------> PlayerManager

PlayerTeam------>PlayerManager

This is a violation of 2NF normal form as there exists a partial depency here.

**3NF**

A single attribute primary key PlayerID can determine all prime/non-prime attributes and no non prime attribute determines another non prime attribute .Thus there are no transitive dependecies. Since the given relation is also in 2NF, the relation is also in 3NF.

Suppose we add another attribute called PlayerWins(number of wins the player has). PlayerWins is determined by PlayerTeam, and not PlayerID

PlayerID----> PlayerTeam

PlayerTeam---🡪PlayerWins

The above is a transitive dependecy. In this case, the relation won’t be in 3NF anymore.

**PLAYER\_STAT**

**FDs**

PlayerID -----> PlayerPosition, Appearances, GoalsScored, YellowCards, RedCards, Suspensions

**1NF**

Each attribute consists of atomic values and each value is single valued. Hence the given relation is in 1NF.

If we add, for example attribute suspension\_type, then 1NF value doesn’t hold good as suspension\_type is multivalued.

**2NF**

A single attribute primary key PlayerID can determine all prime and non-prime attributes, thus we don’t have any partial dependecies. Since the given relation is in 1NF and has no partial dependencies, the relation is also in 2NF.

Violations may occur if we had primary key (KitNo, Team) instead of PlayerID because player position can be determined by KitNo. This would lead to partial dependency.

**3NF**

A single attribute primary key PlayerID can determine all prime/non-prime attributes and no non prime attribute determines another non prime attribute. Thus there are no transitive dependecies. Since the given relation is also in 2NF, the relation is also in 3NF.

Suppose we add another attribute called sending\_offs( 2 yellow cards or 1 red card in a match sends off the player). Thus in this case number of red cards determine the number of sending offs.

PlayerID----->Position, GoalsScored, YellowCards, RedCards

RedCards----->SendingOffs

The above is a transitive dependecy. In this case, the relation won’t be in 3NF anymore.

**TEAM**

**FDs**

TeamName-----> PhoneNo, City

PhoneNo----> TeamName, City

**1NF**

Each attribute consists of atomic values and each value is single valued. Hence the given relation is in 1NF.

Violations may occur if the team has more than one phone no. But in this case only one phone no. is recorded.

**2NF**

A single attribute primary key TeamName can determine all prime and non-prime attributes, thus we don’t have any partial dependecies. Since the given relation is in 1NF and has no partial dependencies, the relation is also in 2NF.

If the team names are not unique, we could use (TeamName, Address) as primary key. But in this case, city depends only on addres.

(TeamName, Address)----->City

Address---->City

This is violation of 2NF as there exists a partial dependency

**3NF**

A single attribute primary key TeamName can determine all prime/non-prime attributes and no non prime attribute determines another non prime attribute. Thus there are no transitive dependecies. Since the given relation is also in 2NF, the relation is also in 3NF.

**TEAM\_STAT**

**FDs**

TeamName ------> Wins, Draws, Losses, Points, GoalDiff , LeaguePosition

LeaguePosition ---------> TeamName, Wins, Draws, Losses, Points, GoalDiff

**1NF**

Each attribute consists of atomic values and each value is single valued. Hence the given relation is in 1NF.

**2NF**

A single attribute primary key TeamName(also foreign key) can determine all prime and non-prime attributes, thus we don’t have any partial dependecies. Since the given relation is in 1NF and has no partial dependencies, the relation is also in 2NF.

If team name is not unique, we could use (TeamName, LeaguePosition) as primary key because no 2 teams can have the same league position. But in this case Points depends on league position.

TeamName, LeaguePosition ------> Points

LeaguePosition ---------> Points

This is a partial dependency and the relation will no longer be in 2NF.

**3NF**

A single attribute primary key TeamName(also foreign key) can determine all prime/non-prime attributes and no non prime attribute determines another non prime attribute. Thus there are no transitive dependecies. Since the given relation is also in 2NF, the relation is also in 3NF.

If we add another attribute called isInRelegation which would have dependencies

TeamName---->LeaguePosition

LeaguePosition-----> isInRelegation

This is a transitive dependency and the relation will no longer be in 3NF.

**STADIUM**

**FDs**

StadiumName ------> Team, Address, Capacity, TurfType, FifaCertification

Team -------> StadiumName

Address-------> StadiumName

**1NF**

Each attribute consists of atomic values and each value is single valued. Hence the given relation is in 1NF.

If a stadium has a dual pitch, we would have a multivalued TurfType attribute, thus violating 1NF.

**2NF**

A single attribute primary key StadiumName can determine all prime and non-prime attributes, thus we don’t have any partial dependecies. Since the given relation is in 1NF and has no partial dependencies, the relation is also in 2NF.

If stadium name is not unique, we could use (StadiumName, Address) as primary key because no 2 stadiums can have the same address.

(StadiumName, Address)------>Capacity, TurfType, FifaCertification

Address------>Team

This is a partial dependency and the relation will no longer be in 2NF.

**3NF**

A single attribute primary key StadiumName can determine all prime/non-prime attributes and no non prime attribute determines another non prime attribute. Thus there are no transitive dependecies. Since the given relation is also in 2NF, the relation is also in 3NF.

**MANAGER**

**FDs**

CoachingLicenseNo --------> MgrName, MgrTeam, MgrDOB, MgrNationality

MgrTeam ------> MgrName

**1NF**

Each attribute consists of atomic values and each value is single valued. Hence the given relation is in 1NF.

**2NF**

A single attribute primary key CoachingLicenseNo can determine all prime and non-prime attributes, thus we don’t have any partial dependecies. Since the given relation is in 1NF and has no partial dependencies, the relation is also in 2NF.

If we removed CoachingLicenseNo, we could use (MgrName, MgrTeam) as primary key. Adding TeamWins, we get dependencies

(MgrName, MgrTeam)------> TeamWins

MgrTeam ------> TeamWins

This is a partial dependency and the relation will no longer be in 2NF.

**3NF**

A single attribute primary key CoachingLicenseNo can determine all prime/non-prime attributes and no non prime attribute determines another non prime attribute. Thus there are no transitive dependecies. Since the given relation is also in 2NF, the relation is also in 3NF.

**MATCH**

**FDs**

MatchID------> MatchWeek, HomeTeam, AwayTeam, MDateTime, RefereeID

HomeTeam, AwayTeam----> MatchID, MatchWeek

HomeTeam, MDateTime -------> AwayTeam, MatchWeek, MatchID, RefereeID

**1NF**

Each attribute consists of atomic values and each value is single valued. Hence the given relation is in 1NF.

**2NF**

A single attribute primary key MatchID can determine all prime and non-prime attributes, thus we don’t have any partial dependecies. Since the given relation is in 1NF and has no partial dependencies, the relation is also in 2NF.

Removing MatchID, we can make (HomeTeam, DateTime). In this case,

HomeTeam, MDateTime -------> MatchWeek

MDateTime-------> MatchWeek

This is a partial dependency and the relation will no longer be in 2NF.

**3NF**

A single attribute primary key CoachingLicenseNo can determine all prime/non-prime attributes and no non prime attribute determines another non prime attribute. Thus are no transitive dependecies. Since the given relation is also in 2NF, the relation is also in 3NF.

**MATCH\_STAT**

**FDs**

MatchID------> HomeTeamGoals, AwayTeamGoals, Attendance

**1NF**

Each attribute consists of atomic values and each value is single valued. Hence the given relation is in 1NF.

**2NF**

A single attribute primary key MatchID(foreign key) can determine all prime and non-prime attributes, thus we don’t have any partial dependecies. Since the given relation is in 1NF and has no partial dependencies, the relation is also in 2NF.

**3NF**

A single attribute primary key MatchID(foreign key) can determine all prime/non-prime attributes and no non prime attribute determines another non prime attribute. Thus there are no transitive dependecies. Since the given relation is also in 2NF, the relation is also in 3NF.

If we had another attribute called outcome, the functional dependencies in this case would be

MatchID-----> HomeTeamGoals, AwayTeamGoals

HomeTeamGoals, AwayTeamGoals---->Outcome

The above is a transitive dependency and the relation is no longer in 3NF.

**REFEREE**

**FDs**

RefereeLicenseNo ------>RefName, RefDOB, RefNationality

**1NF**

Each attribute consists of atomic values and each value is single valued. Hence the given relation is in 1NF.

If we add another attribute called Matches\_officiated, it would be a multivalued attribute as a referee can officiate more than one match. This is a violation of 1NF.

**2NF**

A single attribute primary key RefereeLicenseNo can determine all non-prime attributes, thus we don’t have any partial dependecies. Since the given relation is in 1NF and has no partial dependencies, the relation is also in 2NF.

**3NF**

A single attribute primary key RefereeLicenseNo can determine all prime/non-prime attributes and no non prime attribute determines another non prime attribute. Thus there are no transitive dependecies. Since the given relation is also in 2NF, the relation is also in 3NF.

**BROADCASTER**

**FDs**

Region ------> BroadcasterID

BroadcasterID------> BroadcasterName

**1NF**

Each attribute consists of atomic values and each value is single valued. Hence the given relation is in 1NF.

If we add another attribute called Matches\_Broadcasted, it would be a multivalued attribute as a referee can officiate more than one match. This is a violation of 1NF.

**2NF**

A single attribute primary key Region can determine all prime and non-prime attributes, thus we don’t have any partial dependecies. Since the given relation is in 1NF and has no partial dependencies, the relation is also in 2NF.

**3NF**

A non key attribute BroadcasterID determining another non key attribute BroadcasterName. Primary key is functionally determines only BroadcasterID.

Thus we decompose the table into:-

i)REGIONAL\_BROADCASTER

|  |  |
| --- | --- |
| Region | BroadcasterID |

ii) BROADCASTER\_INFO

|  |  |
| --- | --- |
| BroadcasterID | BroadcasterName |

The above relations are in 3NF.

Dependecies of the original relation are preserved thus this decomposition is dependency preserving.

Test for lossless Join Property:

Decomposition : REGIONAL\_BROADCASTER(Region, BroadcasterID),

BROADCASTER\_INFO(BroadcasterID, BroadcasterName)

F+ = {Region ------> BroadcasterID

BroadcasterID------> BroadcasterName}

|  |  |  |  |
| --- | --- | --- | --- |
|  | Region | BroadcasterID | BroadcasterName |
| REGIONAL\_BROADCASTER | α | α | α |
| BROADCASTER\_INFO |  | α | α |

Since row REGIONAL\_BROADCASTER has all α in columns we have a **lossless decompostion**, i.e if we try to join the above two able we get original table BROADCASTER.

**All the table are in BCNF as all table are in 3NF and no non-prime attribute determines a prime attribute.**

**All table are in 4NF because** **as all table are in BCNF and there are no multivalued dependencies.**

**DDL**

**Note: The relations MATCH and MATCH\_STAT are changed to GAME and GAME\_STAT because of reserved keyword match in MSQL**

CREATE TABLE TEAM(

TeamName VARCHAR(30) NOT NULL,

City VARCHAR(30) NOT NULL,

PhoneNo VARCHAR(15) NOT NULL UNIQUE,

CONSTRAINT TEAMPK

PRIMARY KEY (TeamName)

);

CREATE TABLE PLAYER(

PlayerID CHAR(6) NOT NULL,

PlayerName VARCHAR(30) NOT NULL,

PlayerDOB DATE NOT NULL,

PlayerNationality VARCHAR(30) NOT NULL,

PlayerTeam VARCHAR(30),

KitNo INT NOT NULL,

UNIQUE(KitNo,PlayerTeam),

CONSTRAINT PPK

PRIMARY KEY (PlayerID),

CONSTRAINT PFK

FOREIGN KEY (PlayerTeam) REFERENCES TEAM(TeamName)

ON DELETE SET NULL ON UPDATE CASCADE,

CHECK(1<=KitNo AND KitNo<=99)

);

CREATE TABLE PLAYER\_STAT(

PlayerID CHAR(6) NOT NULL,

PlayerPosition VARCHAR(5) NOT NULL,

Appearances INT DEFAULT 0,

GoalsScored INT DEFAULT 0,

RedCards INT DEFAULT 0,

YellowCards INT DEFAULT 0,

Suspensions INT DEFAULT 0,

CONSTRAINT PSPK

PRIMARY KEY (PlayerID),

CONSTRAINT PSFK

FOREIGN KEY (PlayerID) REFERENCES PLAYER(PlayerID)

ON DELETE CASCADE ON UPDATE CASCADE,

CHECK(GoalsScored>=0)

);

CREATE TABLE TEAM\_STAT(

TeamName VARCHAR(30) NOT NULL,

Wins INT NOT NULL,

Draws INT NOT NULL,

Losses INT NOT NULL,

GoalDiff INT DEFAULT 0,

Points INT NOT NULL,

LeaguePosition INT NOT NULL UNIQUE,

CONSTRAINT TSPK

PRIMARY KEY(TeamName),

CONSTRAINT TSFK

FOREIGN KEY(TeamName) REFERENCES TEAM(TeamName)

ON DELETE CASCADE ON UPDATE CASCADE,

CHECK(1<=LeaguePosition<=20)

);

CREATE TABLE STADIUM(

StadiumName VARCHAR(15) NOT NULL,

Team VARCHAR(15) NOT NULL UNIQUE,

Address VARCHAR(50) NOT NULL UNIQUE,

Capacity INT DEFAULT 40000,

Turftype VARCHAR(10) NOT NULL,

FifaCertification CHAR(1) NOT NULL,

CONSTRAINT SPK

PRIMARY KEY(StadiumName),

CONSTRAINT SFK

FOREIGN KEY(Team) REFERENCES TEAM(TeamName)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE MANAGER(

ManagerName VARCHAR(30) NOT NULL,

MgrTeam VARCHAR(30) NOT NULL UNIQUE,

CoachingLicenseNo char(6) NOT NULL,

MgrDOB DATE NOT NULL,

MgrNationality VARCHAR(30) NOT NULL,

CONSTRAINT MGRPK

PRIMARY KEY(CoachingLicenseNo),

CONSTRAINT MGRFK

FOREIGN KEY(MgrTeam) REFERENCES TEAM(TeamName)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE REFEREE(

RefereeName VARCHAR(30) NOT NULL,

RefereeLicenseNo char(6) NOT NULL,

RefDOB DATE NOT NULL,

RefNationality VARCHAR(30) NOT NULL,

CONSTRAINT RPK

PRIMARY KEY(RefereeLicenseNo)

);

CREATE TABLE GAME(

MatchID char(5) NOT NULL,

MatchWeek INT NOT NULL,

HomeTeam VARCHAR(30) NOT NULL,

AwayTeam VARCHAR(30) NOT NULL,

MDateTime DateTime NOT NULL,

RefID CHAR(6) NOT NULL,

UNIQUE(HomeTeam, MDateTime),

CONSTRAINT MPK

PRIMARY KEY(MatchID),

CONSTRAINT MFK

FOREIGN KEY(HomeTeam) REFERENCES TEAM(TeamName)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(AwayTeam) REFERENCES TEAM(TeamName)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(RefID) REFERENCES REFEREE(RefereeLicenseNo)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE GAME\_STAT(

MatchID char(5) NOT NULL,

HomeTeamGoals INT NOT NULL,

AwayTeamGoals INT NOT NULL,

Attendance INT,

CONSTRAINT MSPK

PRIMARY KEY(MatchID),

CONSTRAINT MSFK

FOREIGN KEY(MatchID) REFERENCES GAME(MatchID)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE BROADCASTER\_INFO(

BroadcasterID char(6) NOT NULL,

BroadcasterName VARCHAR(30) NOT NULL,

PRIMARY KEY(BroadcasterID)

);

CREATE TABLE REGIONAL\_BROADCASTER(

Region VARCHAR(30) NOT NULL,

BroadcasterID char(6),

CONSTRAINT RBPK

PRIMARY KEY(REGION),

CONSTRAINT RBFK

FOREIGN KEY(BroadcasterID) REFERENCES BROADCASTER\_INFO(BroadcasterID)

ON DELETE SET NULL ON UPDATE CASCADE

);

**SQL Triggers**

1. **To ensure the number of teams playing the league does not exceed 20.**

DELIMITER $$

CREATE TRIGGER exceed\_team\_count

BEFORE INSERT

ON team FOR EACH ROW

BEGIN

DECLARE teamCount INT;

SELECT COUNT(\*)

INTO teamcount

FROM team;

IF teamcount>20 THEN

SET new.TeamName = NULL;

END IF;

END$$

DELIMITER ;

1. **To update the team\_stat table automatically as and when the match\_stat table is updated.**

DELIMITER $$

CREATE TRIGGER update\_team\_stat

AFTER INSERT

ON game\_stat FOR EACH ROW

BEGIN

DECLARE HomeTeam VARCHAR(30);

DECLARE AwayTeam VARCHAR(30);

SELECT g.HomeTeam, g.AwayTeam

INTO HomeTeam, AwayTeam

FROM game as g

WHERE g.MatchID = new.MatchID;

IF new.HomeTeamGoals>new.AwayTeamGoals THEN

UPDATE team\_stat

SET Wins = Wins + 1

WHERE TeamName = HomeTeam;

UPDATE team\_stat

SET Points = Points + 3

WHERE TeamName = HomeTeam;

UPDATE team\_stat

SET Losses = Losses + 1

WHERE TeamName = AwayTeam;

END IF;

IF new.HomeTeamGoals<new.AwayTeamGoals THEN

UPDATE team\_stat

SET Wins = Wins + 1

WHERE TeamName = AwayTeam;

UPDATE team\_stat

SET Points = Points + 3

WHERE TeamName = AwayTeam;

UPDATE team\_stat

SET Losses = Losses + 1

WHERE TeamName = HomeTeam;

END IF;

IF new.HomeTeamGoals=new.AwayTeamGoals THEN

UPDATE team\_stat

SET Draws = Draws + 1

WHERE TeamName = AwayTeam;

UPDATE team\_stat

SET Points = Points + 1

WHERE TeamName = AwayTeam;

UPDATE team\_stat

SET Points = Points + 1

WHERE TeamName = HomeTeam;

UPDATE team\_stat

SET Draws = Draws + 1

WHERE TeamName = HomeTeam;

END IF;

UPDATE team\_stat

SET GoalDiff = GoalDiff + (new.HomeTeamGoals - new.AwayTeamGoals)

WHERE TeamName = HomeTeam;

UPDATE team\_stat

SET GoalDiff = GoalDiff + (new.AwayTeamGoals - new.HomeTeamGoals)

where TeamName = AwayTeam;

UPDATE team\_stat T

JOIN (

SELECT \*,

ROW\_NUMBER() OVER (ORDER BY Points DESC, GoalDiff DESC, TeamName ASC) AS RN

FROM team\_stat)T1

ON T1.TeamName=T.TeamName

SET T.LeaguePosition=RN ;

END$$

DELIMITER ;

1. **To make sure that attendance of a match does not exceed stadium capacity. Incase it does, the attendance in wrong and updated as NULL.**

DELIMITER $$

CREATE TRIGGER attendance\_check

BEFORE INSERT

ON game\_stat FOR EACH ROW

BEGIN

DECLARE HomeTeam Varchar(50);

DECLARE capacity INT;

SELECT g.HomeTeam

INTO HomeTeam

FROM game as g

WHERE g.MatchID = new.MatchID;

SELECT s.Capacity

INTO capacity

FROM stadium as s

WHERE s.Team = HomeTeam;

if capacity<new.Attendance then

set new.Attendance = NULL;

end if;

END$$

DELIMITER ;

**SQL Queries**

1)**Find the manager name of the youngest player in the league.**

SELECT ManagerName

FROM manager

WHERE ManagerName in (

SELECT ManagerName

FROM manager

WHERE MgrTeam in (

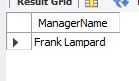
SELECT PlayerTeam

FROM player

WHERE PlayerDOB = (SELECT MAX(PlayerDOB) FROM player)

)

);

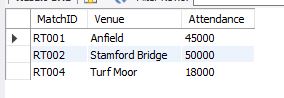


2)**Find venue and attendance of the scheduled matches.**

SELECT game.MatchID, StadiumName as Venue, Attendance

FROM ((stadium RIGHT OUTER JOIN game ON Team = HomeTeam)

RIGHT OUTER JOIN game\_stat ON game.MatchID = game\_stat.MatchID);

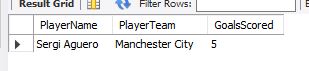


3)**Name, team name and number of goals scored by the oldest player.**

SELECT PlayerName, PlayerTeam, GoalsScored

FROM (player LEFT OUTER JOIN player\_stat ON player.PlayerID = player\_stat.PlayerID)

WHERE PlayerDOB = (SELECT MIN(PlayerDOB) FROM player);



4)**Broadcaster name in Australia.**

SELECT BroadcasterName

FROM broadcaster\_info

WHERE BroadcasterID IN ( SELECT BroadcasterID

FROM regional\_broadcaster

WHERE Region = 'Australia');

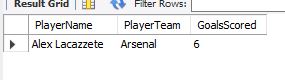


5)**Find name and team of top goal scorer.**

SELECT PlayerName, PlayerTeam, GoalsScored

FROM ( player LEFT OUTER JOIN player\_stat ON player.PlayerID = player\_stat.PlayerID )

WHERE GoalsScored = (SELECT MAX(GoalsScored) FROM player\_stat);



6)**Find total goals scored by Leicester city at home.**

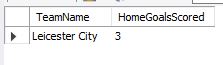
SELECT TeamName, SUM(HomeTeamGoals) AS HomeGoalsScored

FROM ( (team\_stat RIGHT OUTER JOIN game ON team\_stat.TeamName = game.HomeTeam)

RIGHT OUTER JOIN (SELECT MatchID, HomeTeamGoals FROM game\_stat)gs ON game.MatchID = gs.MatchID)

WHERE TeamName = 'Leicester City'

GROUP BY TeamName;



**CONCLUSION**

1. This system is capable of updating team statistics, i.e team’s wins, losses, draws, points, league position as and when the game results are inserted in real time.
2. It makes sure no more than 20 teams can participate in the league.
3. It is also capable of inserting and deleting player information as and when they leave or join during the course of the season.
4. It keeps track of player’s statistics i.e the number of goals score, playing position, red cards, yellow cards, suspensions etc.
5. The user can easily find out information like youngest player, top goalscorer, team at the top of the league, manager with most wins etc.
6. It can be further used for other football leagues like the bundesliga, Seria A, La liga etc.

Limitation

The system is not capable of recording and finding out who scored the goals and at which instance during a match. Also the system cannot update and record of possesion of ball during the course of the match.

The system can record league information for only **one season**.

Future enhancement

The system can be further enhanced to deliver real time score during the course of the games. Other match statistics can also be recorded like ball player heat maps, shots taken etc.

The system can be developed to record multiple league season information.