Entities

Users

Attributes: UserId (PK), UserName, Email

Assumptions: Represents users such as fans or administrators. Each user has a unique ID which can uniquely identify a user.

Reason for Entity Modeling: This entity is modeled to manage user personal data. This can be extended for user authentication, preferences, and other functionalities that might be needed in the future.

Cardinality: Users can like multiple teams and players, and each team and player can be liked by multiple users, indicating a many-to-many relationship with both teams and players through likes.

Teams

Attributes: TeamName (PK), GoalDifference, Wins, Losses

Assumptions: Each TeamName can uniquely represent a team, ensuring no two teams share the same name. This is used to represent various football teams along with their performance metrics and to establish a link with players.

Reason for Entity Modeling: This entity captures comprehensive details about teams, which are essential for analyzing team performance and statistics. Modeling teams as a separate entity allows for storing specific performance metrics and linking with players and match game history.

Cardinality: Each team can have multiple players, establishing a one-to-many relationship with players. Additionally, each team has participated in multiple games giving a many-many relationship with match game history given that each match comprises of two teams.

Players

Attributes: PlayerId (PK), PlayerName, Team, Appearances ,Goals Scored ,Cards,CleanSheets,Assists,Position

Assumptions: Represents individual players and their team affiliations with their background. Each Player can be uniquely identified by a PlayerId that cannot be shared.

Reason for Entity Modeling: This entity is modeled to manage player details such as their personal information and their affiliation with teams. It also links with game history for performance tracking, enabling detailed analysis of each player's contributions.
Cardinality: Players are part of only one team at a time, ensuring a many-to-one relationship with teams as a team is composed of multiple players.A user can like multiple players and a player can be liked by multiple players indicating a many-many relationship with users through likes.

Stadiums

Attributes: StadiumName (PK),Capacity, Surface Type, Location
Assumptions: Represents stadiums, each associated with multiple matches that can be uniquely identified by the StadiumName.
Reason for Entity Modeling: This entity stores stadium-related information, which is crucial for the venue details. Modeling stadiums as separate entities allows for storing capacity,surface type,geographic location and linking each stadium with matches.

Cardinality: One-to-many with MatchGameHistory as a stadium can be used for multiple matches throughout the season,but a particular match can be played at only one stadium.

MatchGameHistory

Attributes: GameId (PK),Attendance,HomeTeam,AwayTeam,HomeTeamGoals ,AwayTeamGoals,Stadium
Assumptions: Each game is uniquely identifiable by its GameId, ensuring precise tracking of match events.The game captures critical competitive metrics such as team performance in terms of goals and match attendance.Matches are linked to specific stadiums, reflecting the venue's influence on game dynamics.
Reason for Entity Modeling: This entity captures all relevant details about each match, including attendance, team performance, and the venue.By modeling this as a separate entity, the database can effectively track and analyze detailed match data over time, providing insights into team performances, fan engagement, and venue utilization.

Cardinality: Many-to-many with Teams (through HomeTeam and AwayTeam): A single match involves one home team and one away team. Teams participate in multiple matches too. Many-to-one with Stadiums as A particular game is played at a single stadium. However multiple games can be played at the same venue.

Normalization

The database schema adheres to BCNF which avoids redundant data and preserves the data integrity by imposing stricter rules than 3NF:

Users(UserId,Username,Email) has FD:

UserId → Username, Email

Teams(TeamName, GoalDifference, Wins, Losses) has FD:

TeamName → GoalDifference, Wins, Losses

Players
(PlayerId,PlayerName,Team,Appearances,GoalsScored,Cards,CleanSheets,Assists,Position) has FD:

PlayerId→PlayerName,Team,Appearances,GoalsScored,Cards,CleanSheets,Assists,Position

Stadiums(StadiumName, Capacity, Surface Type, Location) has FD:

StadiumName → ,Capacity, Surface Type, Location

MatchGameHistory(GameId ,Attendance, HomeTeam, AwayTeam, HomeTeamGoals, AwayTeamGoals, Stadium) has FD:

GameId → Attendance, HomeTeam, AwayTeam, HomeTeamGoals, AwayTeamGoals, Stadium

All above functional dependencies have left side as super key(primary key) and are non trivial proving that we follow BCNF.

Logical Relational Schema

Users
Users(UserId: VARCHAR(50) [PK], UserName: VARCHAR(50), Email: VARCHAR(100))

Teams
Teams(TeamName: VARCHAR(255) [PK], GoalDifference: INT, Wins: INT, Losses: INT)

Players
Players(PlayerId: INT [PK], PlayerId INT,PlayerName VARCHAR(255),Team VARCHAR(255) [FK to Teams.TeamName],Appearances INT,GoalsScored INT,Cards INT,CleanSheets INT, Assists INT,Position VARCHAR(255),

Stadiums
Stadiums(StadiumName: VARCHAR(255) [PK], Capacity:INT ,SurfaceType: VARCHAR(255) Location:VARCHAR (255))

MatchGameHistory
MatchGameHistory(GameId INT [PK] , Attendance INT, HomeTeam VARCHAR(255) [FK to Teams.TeamName], AwayTeam VARCHAR(255][FK to Teams.TeamName], HomeTeamGoals INT, AwayTeamGoals INT, Stadium VARCHAR(255) [FK to Stadiums.StadiumName])

User's Liked Players
UserLikedPlayers(UserId1: INT [PK] [FK to Users.UserId], PlayerId1: [PK] [FK to Players.PlayerId])

User's Liked Teams
UserLikedTeams(UserId1: INT [PK] [FK to Users.UserId], TeamId1: INT [PK] [FK to Teams.TeamName])

Teams Participated in Games

TeamsParticipatedinGames(TeamName1: INT [PK] [FK to Teams.TeamName], TeamName2: INT [PK] [FK to Teams.TeamName],GameId1: [PK] [FK to MatchGameHistory.GameId])