Databases
Informatics and Computing Engineering Department
Faculty of Engineering, University of Porto

Portuguese Football League

A Database System

´

Duarte Gonçalves up202108772

Gonçalo Miranda up202108773

Marco Vilas Boas up202108774

2LEIC01, Group 101

November 2022

Index

Figure Index

Context

Football is the most famous sport in the world, with 275 million active players around the globe. With its origins in Britain in the 19th century, this sport is a part of our lives and, therefore, it must be organized and regulated by strict rules to keep everything in order.

Databases can help us keep track of everything that happens inside a football league/competition. To accomplish this, we need to divide the league into elements and define explicit relations between all of them. This way we can retrieve data from any point of the league. Our aim with this project is to provide a database capable of describing the Portuguese Football League.

The Portuguese Football League (PFL) is a competition between 18 teams, which play each other 2 times during the year, corresponding to 34 rounds during 1 season. Each round 9 games happen, and after each game the league scoreboard is updated with the result: 3 points to a team for a win, 1 point for draws and 0 for losses. Each game is hosted in a stadium and many events happen, being the main one the goal.

According to the description given in the paragraph above, we have designed a UML model to describe all the elements that compose the FPL and the way they relate to each other. We also provide a relational model, its functional dependencies and restrictions.

The point of these schemes is to create a SQLite database system and that is why certain things aren´t explicitly implemented, for example the standing after each round, or the teams in risk of being demoted. These are not present in the UML/Relational Model because they are ‘calculable’, that is, starting from the data we have in the relations/tables we can calculate all these things.

Conceptual Modeling

UML Diagram

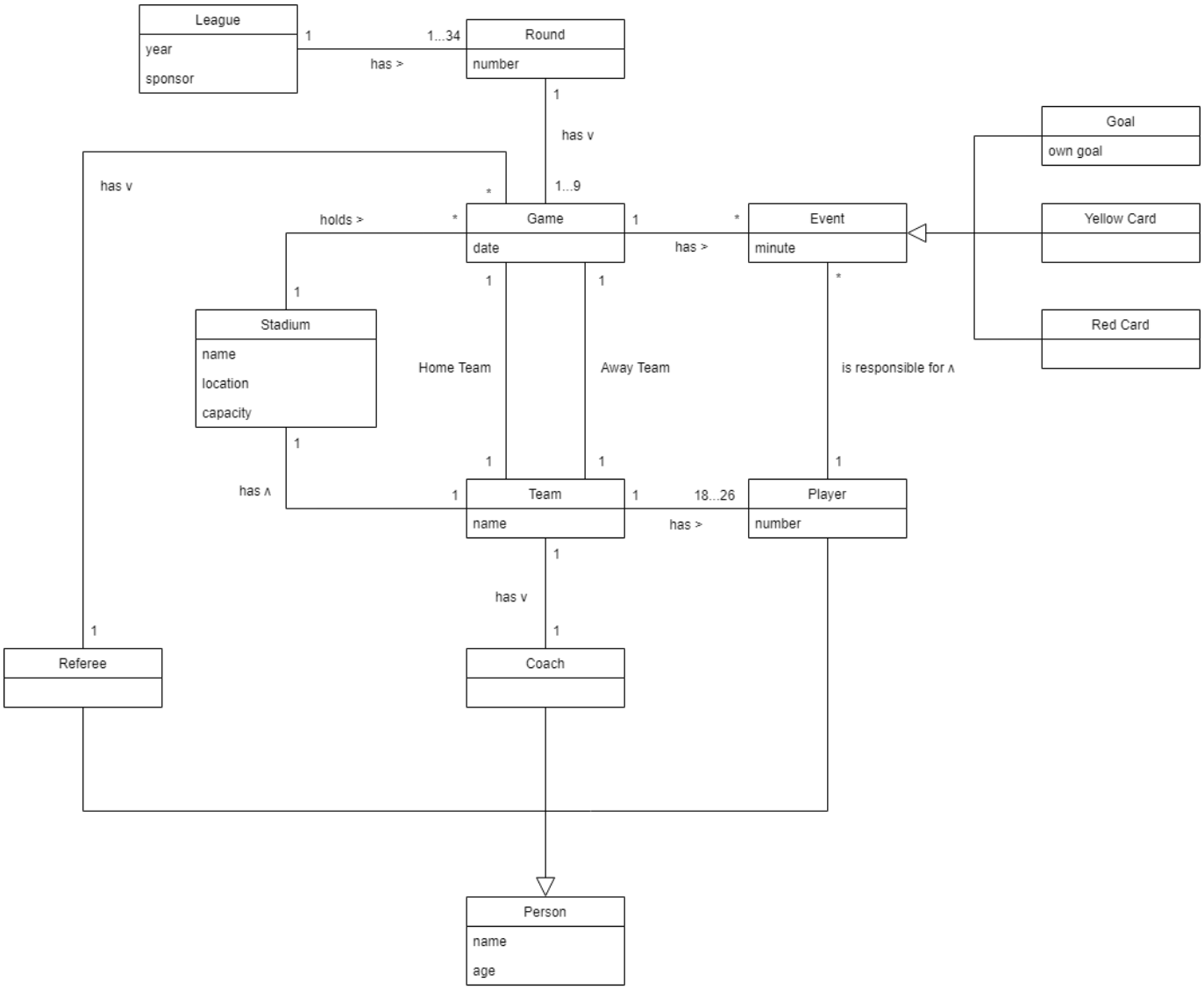

Figure 1 - UML Diagram.

Class Definitions and Restrictions

Class	Definition	Restrictions
League	Defines the league being played. It has as its attributes the season's end year and main sponsor.	None
Round	Represents each round that occurs in a given league. It is characterized by its number, ranging from 1 to 34.	a) $1 \leq number \leq 34$
Game	Defines a game, played by the two, visited and visitor teams, on a given date.	None
Team	Describes a football team, as a collective of people, coach and players. It has its own, unique name and stadium.	a) name is unique
Stadium	Defines a stadium, where different games take place. It is associated with a team, and characterized by its name, location, and capacity.	a) name is unique b) capacity > 0
Person	Represents a person, which could be either a player, a coach, or a referee. This person is described by name and age.	a) age ≥ 16
Player	Defines a football player, part of a given team. This person is responsible for scoring goals, receiving either red or yellow cards, in matches.	a) $1 \leq number \leq 99$
Coach	Defines the football coach of a team.	None
Referee	Defines the referee, a sports official responsible for conducting a game between two teams.	None
Event	Represents a generalization of an event which takes place during a football match; this can be a goal, a yellow or red card. It has the minute it occurred as an attribute.	a) minute ≥ 0

Goal	Defines a goal scored by a specific player in a certain minute of the match. It can be an own goal.	None
Red Card	Defines a red card shown to a player in a game.	None
Yellow Card	Defines a yellow card shown to a player in a game.	None

Figure 2 - Class Definitions and Restrictions.

Relational Model

In all the explicit relations, the primary key is defined with an underline and the foreign keys have an arrow pointing to the relation they reference.

league (year, sponsor)

round (number, league_year → league)

game (id, date, stadium_name → stadium, referee_id → referee, home_team_name → team, away_team_name → team, round_number → round)

team (name, stadium_name)

stadium (name, location, capacity, team_name → team)

person (id, name, age)

player (id → person, number, team_name → team)

coach (id → person, team_name → team)

referee (id → person)

event (id, minute, player_id → player, game_id → game)

goal (id → event, own_goal)

red_card (id → event)

yellow_card (id → event)

Functional Dependencies

Given the structure of our relational model, the non-trivial functional dependencies are mostly of the form:

{pk} → {every other attribute}

Except for some relations that don’t have functional dependencies at all.

Since the domain of each attribute in each relation only contains atomic values and the value of each attribute contains only a single value from its domain, we conclude that the dependencies verify the First Normal Form. Since no not prime attribute functionally depends on a (proper) subset of a candidate key, the Second Normal form is also verified. Since {pk} is a superkey, the Third Normal Form is also verified. Furthermore, we also conclude that the BCNF is verified.

Below is the table listing all such non-trivial functional dependencies:

Relation	**Functional Dependencies**
League	{year} → {sponsor, round_number}
Round	{number} → {game_id}
Game	{id} → {date, stadium_name, referee_id, home_team_name, away_team_name, event_id}
Team	{name} → {stadium_name}
Stadium	{name} → {location, capacity, team_name}
Person	{id} → {name, age}
Player	{id} → {number, team_name}
Coach	{id} → {team_name}
Referee	None
Event	{id} → {minute, player_id}
Goal	{id} → {own_goal}
Red_card	None
Yellow_card	None

Restrictions

In this section we are going to list all restrictions associated with primary and foreign keys and mandatory parameters.

Primary keys and unique values

As stated before, every table will have one attribute which is the primary key, therefore having a primary key constraint. Furthermore, there are some values that may be unique, that is, there can be no other tuple in the table with the same value-attribute. Those values are listed here:

- In table **team**, the attribute stadium_name must be unique.
- In table **stadium**, the attribute team_name must be unique.
- There can only be one **player** with a given number for a given team.

Foreign keys

For all attributes that refer to another table a foreign key constraint must be set. This applies for all foreign keys, which are the parameters followed by a ‘ → ’ in the Relational Model.

Context Restrictions

Following is a list of restrictions for some attributes of certain tables:

- In **league**, year must be greater than 0.
- In **round**, the number must be between 1 and 34, including.
- In **game**, the round_number must also be between 1 and 34, including.
- In **stadium**, the capacity must be greater than 0.
- In **person**, the age must be greater than 0.
- A **player**’s number must be between 1 and 99, including.
- In **goal**, the own goal attribute must be either true or false.

Mandatory Parameters: not null

There are some parameters that are crucial to the database, and therefore cannot have the value **null**. That is the case with all the primary keys and all the foreign keys. The only parameter apart from all those we made mandatory is the **player**’s name.