

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	3
Context	4
Conceptual Modeling	5
Class Definitions and Restrictions	6
Relational Model	7
Functional Dependencies	8
Restrictions	9
Primary keys and unique values	9
Foreign keys	9
Context Restrictions	9
Mandatory Parameters: not null	9

Figure Index

FIGURE 1 - UML DIAGRAM.	6
FIGURE 2 - CLASS DEFINITIONS AND RESTRICTIONS.	7

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

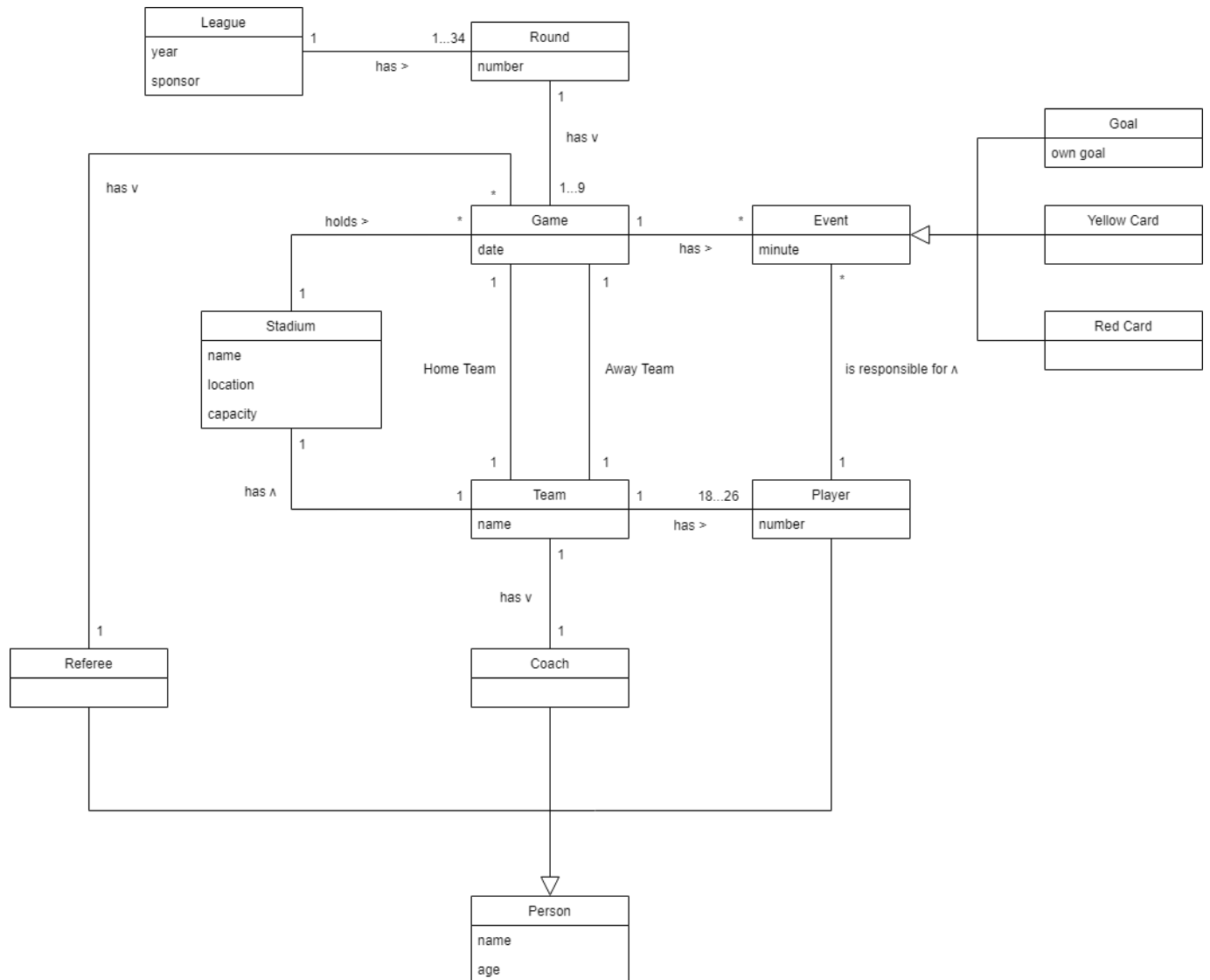


Figure 1 - UML Diagram.

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) $1 \leq \text{number} \leq 34$
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) $1 \leq \text{number} \leq 99$
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:

$\{pk\} \rightarrow \{\text{every other attribute}\}$

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	$\{year\} \rightarrow \{sponsor, round_number\}$
Round	$\{number\} \rightarrow \{game_id\}$
Game	$\{id\} \rightarrow \{date, stadium_name, referee_id, home_team_name, away_team_name, event_id\}$
Team	$\{name\} \rightarrow \{stadium_name\}$
Stadium	$\{name\} \rightarrow \{location, capacity, team_name\}$
Person	$\{id\} \rightarrow \{name, age\}$
Player	$\{id\} \rightarrow \{number, team_name\}$
Coach	$\{id\} \rightarrow \{team_name\}$
Referee	None
Event	$\{id\} \rightarrow \{minute, player_id\}$
Goal	$\{id\} \rightarrow \{own_goal\}$
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 ' \rightarrow ' 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.

Queries

Next, we're going to list the database's queries. The following queries have been implemented based on relevant questions that may arise when using the database, as well as taking this project's guide, where it was specified that certain queries must be implemented. Those are the queries number 4 and number 5.

Query 1

Which players are in "Porto"'s roster?

This interrogation lists the names of all players from the team "Porto" present in the database, as well as their number and age.

Query 2

What teams have played in "Estádio do Dragão" until round 6?

Gets the names of all teams that "Porto" faced, while being the home team, until the 6th round of the championship, regardless of the game's final score.

Query 3

What games were won with a difference of two or more goals?

This query displays the date of all games where the winning team won with more than two goals compared to the loser, as well as their stadium, both teams and their respective scores.

Query 4

What does the classification table for this round look like? Who's first and who's last?

This interrogation lets the user check the championship's classification table for a specific round (in the example, round 6 was chosen, as it is the last round displayed in the database). It displays the name of all the teams playing the championship, their position, their points, and both their scored and conceded goals. The table is ordered by the number of points and, in the case of a draw, it sorts by the amount of goals scored and by the amount of goals conceded.

Query 5

What teams are going to the European competitions? And what teams are in risk of relegation?

Displays the three top teams of the championship, that are the ones that'll go to the European competitions, and the two bottom teams, that are the ones that are in risk of relegation, as well as their points, goals scored and goals conceded, on a specific round (in the example, round 6 was chosen, as it is the last round displayed in the database), ordered by the number of points and, in the case of a draw, it sorts by the amount of goals scored and by the amount of goals conceded.

Query 6

Who's Nuno Almeida? What games did he refer?

Lists the round, date, stadium and both the teams from all the games referred by the referee Nuno Almeida.

Query 7

Who scored to tie the score?

This query displays all the players who scored in draws and their respective teams.

Query 8

Who got 2 or more yellow cards?

Lists the team, the names, the shirt numbers and the number of yellow cards of players who have received 2 or more yellow cards until a certain round (in this case, the results are up to round 4).

Query 9

What's the top 5 scorers of the championship?

Displays the team, the name and number of goals scored by the top 5 scorers of the championship.

Query 10

What's the win rate of teams when playing away?

Displays all the teams in the competition and their respective win rate when playing away.

Query 11

What's the win rate of teams who score the first goal of the game?

Displays every team's name and its win rate when scoring the first goal.

Triggers

The following triggers seek to keep the database coherent, avoiding any malfunctions that may come with wrongly inserted data. The implementation of these triggers was based on relevant problems that may arise when using the database.

Trigger 1

Two yellow cards result in a red card.

Creates a red card as soon as there's two yellow cards to the same player.

Trigger 2

An event must be associated with a player in the game

Before inserting an event, checks if the player associated to that event is from one of the two teams present in the game associated to it. In that case, the insertion is aborted, and an error message is displayed.

Trigger 3

Two teams can't face each other twice in the same round.

Before inserting a game between a pair of teams, this trigger checks if there's already a game where these two teams play each other (regardless of what team will be the home team and what team will be the away team). If so, it aborts the insertion and displays an error message.

Final Considerations

To conclude, we reflected on what went well and on what could've been better. Therefore, we decided to list our reflections on what could be improved:

- More diversity of events – player substitution, for example.
- More attributes – for example, using round as a game attribute instead of a whole class, or implementing a timestamp (minute) as an event attribute.
- Better table and attribute names.
- Better organization on ID-based insertions.
- Better describing events and people involved in events.

Besides that, there are some queries that look less realistic than they should. That comes from the fact that not all games from this season have been inserted into the database, which may create odd situations, like, for example, "Rio Ave" being one of the teams in risk of relegation (using query 5), when they currently stand 10th on the championship's league (only 13 rounds have been played, but more than 6 have been inserted).

Group Self-evaluation

All members contributed fairly to the project's outcome, being everything fairly distributed.