Basics of database systems

Project – Database design: Premier League 22/23

Lappeenranta-Lahti University of Technology LUT
Software Engineering

Basics of database systems
Spring 2023
Veeti Rajaniemi
veeti.rajaniemi@student.lut.fi

TABLE OF CONTENTS

1 DEFINITION

In my project "Premier League 22/23", a database is developed for someone who deals with Premier League statistics and needs to find certain information about it. The user might be for example a scout or a data analyst who analyses this kind of different statistics or even a football fan who just wants to see and handle stats and results in a simple way. The database includes information about Premier League players and teams in season 2022/2023. There are some basic statistics of all players and a bit more advanced shooting and goalkeeping stats. The database also includes results and some other basic information about played matches. There are basic and wage statistics of all teams as well.

I wanted to create this database with a decent amount of data, so I decided to use some open data from the internet. All data used are from the following page: https://fbref.com/en/comps/9/stats/Premier-League-Stats. I tried to choose the most important stats which are reasonable to the database and give important information. I created a playerID and a matchID to all players and matches to make usage of the database simpler. These two and the team names are the most important information in the database.

I also decided to create a Python interface to my project just to make it a bit more interesting. In addition, I used Python for reading and parsing the data and creating the commands for inserting it to the database. With my user interface I wanted to give user some options to search for in different queries. I didn't pay that much attention to the design of the interface and how the results are shown – the most important part was to make an interface which works and gives relevant information. Also, there could be more error handling within the Python code if the user's inputs are invalid, for example.

The following database queries are implemented:

1) List all players from a team. With the user interface the user can choose a team which players will be listed.
2) List all team stats. The basic stats and wage stats are combined in this query.
3) List player's special stats based on the playerID. If player's position is goalkeeper ("GK"), the query will show player's goalkeeping stats. In other cases, the query

will show player's shooting stats. With the user interface I decided that user can search for stats by inserting player's name and team (not a playerID, which will be used in table modification) and then the program will find the correct playerID and list info based on it and the position of a player.

4) List all match results and stats of a team. The results are combined by joining the Team, Match_results and Team_Match tables together so everything is easy to see.
5) List top 10 goal scorers of the season. With the user interface the amount of players listed can be chosen.

There is also a possibility for user to modify the Player table with the interface. It's possible to insert a new player, delete an existing player or change a player's team. The last option could be done when a player transfers to a new club during the season. With modifying, I decided to let the user search for a player by inserting the playerID. When a new player is added, the playerID is one bigger than the current maximum.

2 MODELING

2.1 Concept model

In figure 1 there is the ER model of the designed database. There are six entities in the model and five relationships. Every team has to have at least one player and the maximum number of players is N, so we have a one-to-many relationship. There is one many-to-many relationship in between Team and Match_results. In this case the relation is 2:N (this data is from one season so maximum of N would be 38 if all matches were included), because a team can be included in multiple matches and there are always exactly two teams in one match result. The many-to-many relationship is demonstrated with the query 4. There are three derived attributes, 'annual_wages', 'goals_against_per90min' and 'shots_on_target_percentage', which will be removed. The multivalue 'position' will be reduced to a single value, a preferred position. It's good to note that every player has shooting stats, but only goalkeepers have goalkeeping stats.

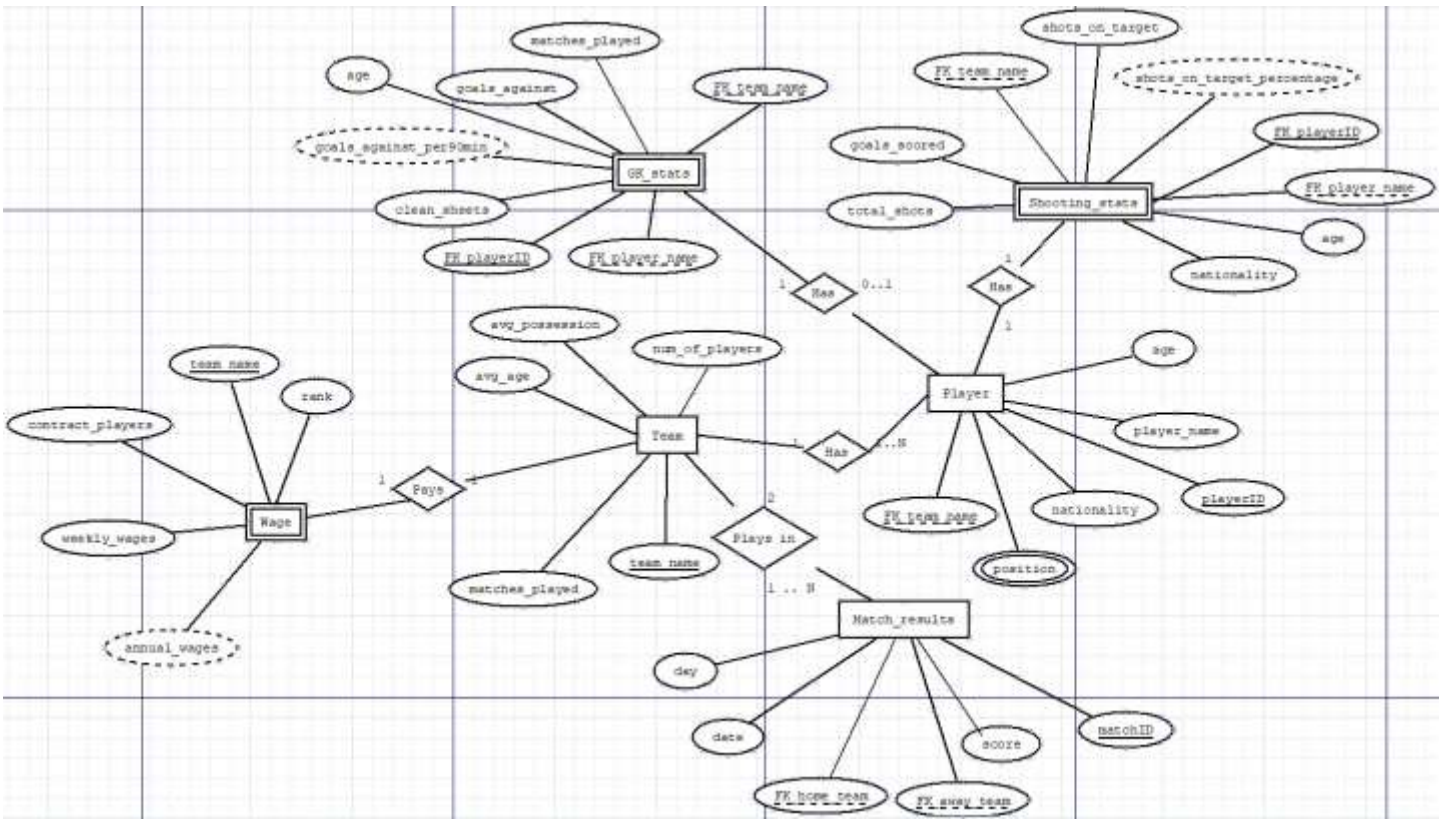

Figure 1: ER model

2.2 Relational model

Figure 2 shows the relational model which was created based on the ER model. The linking table "Team_Match" was created because of the many-to-many relationship between Team and Match_results entities. As planned, the derived attributes are removed and the multivalue is reduced to a single value.

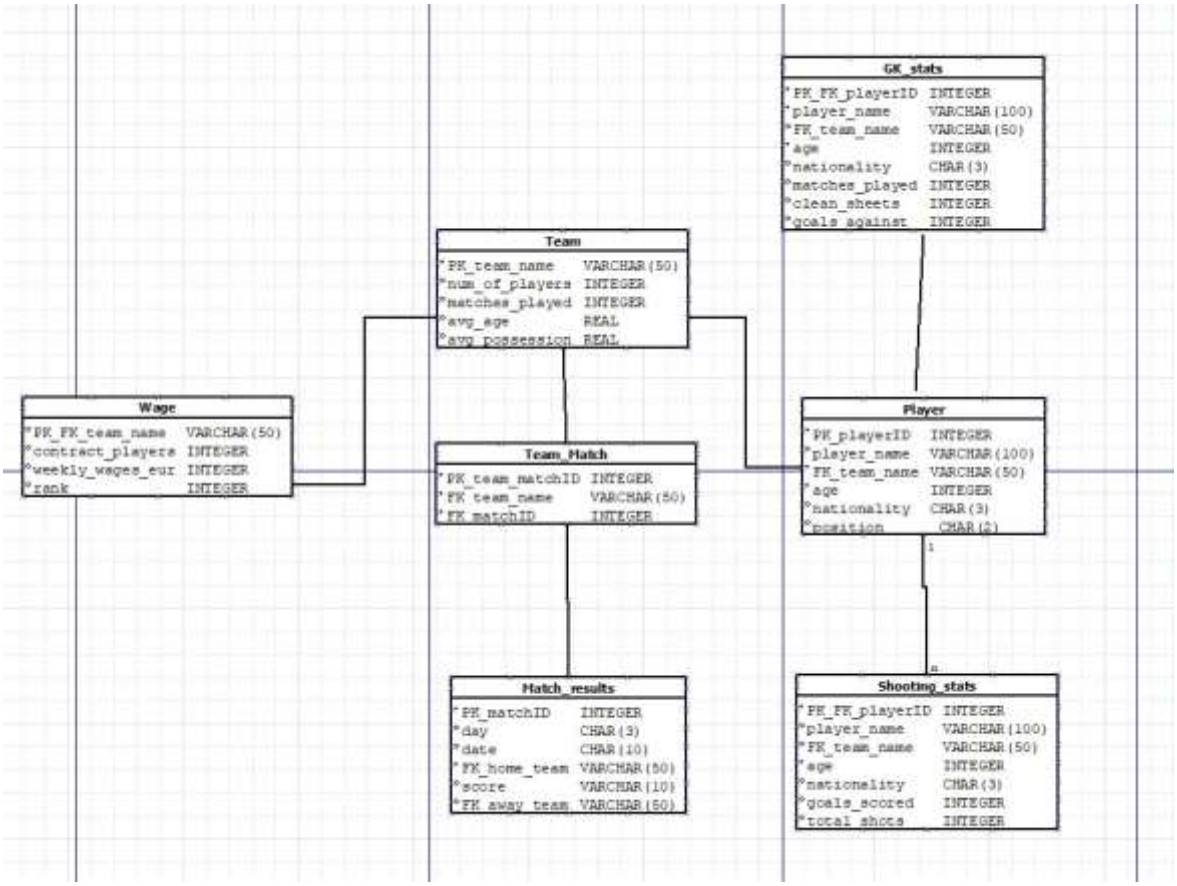

Figure 2: Relational model based on the ER model

3 DATABASE IMPLEMENTATION

3.1 Constraints and indices

During implementation, the following constraints are created for the relations:

- **Team**:
  - team_name NOT NULL
  - avg_possession has to be under 100 (CHECK < 100)
- **Player**:
  - foreign key reference to team
  - playerID, team_name, age NOT NULL
- **GK_stats**:
  - playerID, player_name, team_name, age NOT NULL
  - matches_played DEFAULT 0
  - foreign key reference to Player and Team, with both ON DELETE CASCADE
- **Match_results**:
  - matchID, home_team, away_team NOT NULL
  - 2 foreign key references to Team, with both ON UPDATE CASCADE
- **Shooting_stats**:
  - playerID, age NOT NULL
  - goals_scored, total_shots DEFAULT 0
  - foreign key references to Player (ON DELETE CASCADE) and Team (ON UPDATE CASCADE)
- **Wage**:
  - rank has to be unique and under 21 (UNIQUE CHECK < 21)
  - foreign key reference to Team, ON UPDATE CASCADE
- **Team_match**
  - team_matchID, team_name, matchID NOT NULL
  - foreign key references to Team (ON UPDATE CASCADE) and to Match_results (ON UPDATE CASCADE)

In addition to the integrity constraints listed above, the database will also implement seven different indices. The indices are created because the values are often used in queries or they are used to ordering.

3.2 User interface with Python

In addition to the database, I also made a Python interface as told, made with the sqlite3 -library. The interface is simple, and it demonstrates the 5 queries made. The interface has a menu with 6 options – five queries and an option to modify player data. Every option has its own function which does the work. The user interface is connected to the database in the beginning of the code.