COMP 3005 Term Project

Seyedsajad Hosseini

101205877

Contents

[Conceptual Design 2](#_Toc163761423)

[Reduction to Relation Schemas 6](#_Toc163761424)

[Database Schema Diagram 11](#_Toc163761425)

[Bonus Queries 13](#_Toc163761426)

# Conceptual Design

The process of designing the database and writing the code to load JSON files into the PostgreSQL database began with a thorough analysis of the data format stored within the JSON files. During this phase, careful attention was given to studying the Open Data documentation files, which provided insights into less commonly used attributes. In accordance with the project specifications, to maximize data integration into the database, a decision was made to adhere closely to the structure of the Open Data, resulting in the creation of 39 tables.

This process resulted in the successful identification of the essential entities necessary for constructing a relational database. The main entities and their relations with other entities include:

* Team: a team has many players, one or more manager, participates in zero or more matches and competitions
* Player: a player can have multiple jersey numbers, has zero or more team (this can happen when a player is transferred to another team), plays in one or more positions, and may receive zero or more cards
* Match: a match has many (two) teams, many positions for the players to play in, exactly one referee (the json files do not include assistant referees), exactly one stadium, and zero or more events
* Event: event is anything that happens during a match including passes, shots, dribbles, fouls, etc. An event is performed by a player in a match, therefore, each event has exactly one player and one match.

After identifying the entities within the database, the next crucial step was to map them to their respective tables. This phase was the most challenging part of the project, as it directly impacted the efficiency of queries and the implementation of the json loader. Key challenges encountered during this step included:

* How to store the events?

Each events json file contains thousands of different events characterized by shared and unique attributes. One approach to store these events involves creating a single large event table with attributes covering all event types. While straightforward in its implementation, this approach leads to a multitude of null values. This happens because, for example a pass event doesn’t have attributes such as card type or statsbombXG. Consequently, such a table can result in poor query performance.

Alternatively, a second approach includes establishing individual tables for each event type. While this method yields better query performance, it generates numerous tables, primarily composed of common attributes alongside a handful of unique ones. Despite its efficiency, this approach creates unnecessarily large number of, still, large tables, which was considered undesirable for this project.

The third approach, employed in the submitted database, involves creating a central “events” table housing shared attributes alongside several smaller tables containing unique attributes for each event type. In this approach, some match events possess no distinct attributes and thus do not require separate tables. However, for passes and shots, it was decided to create dedicated events tables to include both shared and unique attributes. This was a deliberate move to further improve the efficiency of queries pertaining to passes and events. Apart from passes and shots, attributes for all other events are distributed between the central events table and type-specific tables. It's worth noting that certain attributes in type-specific tables aren't exclusive to a single event type. However, as these attributes aren't shared by "all" events, they have been relocated to their respective type-specific event tables.

* How to store attributes with ids?

Some attributes in the json files contain both an id and a name, such as country, pass type, foul type, and event outcome. When storing these attributes in tables, there are two approaches:

1. Storing only the attribute name.
2. Using an additional table for each attribute to store both the id and the name, and then referencing the id in other tables.

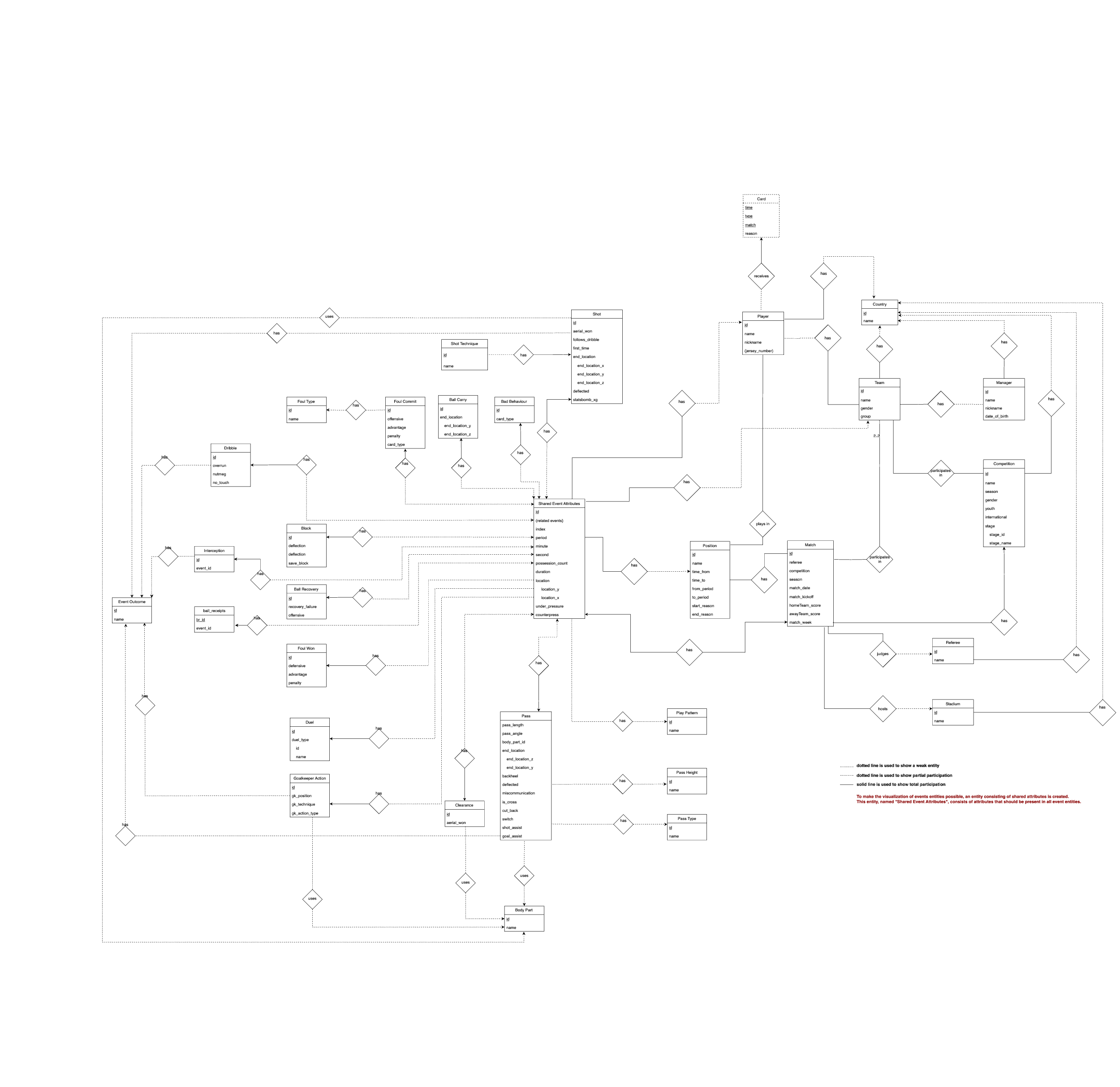
For instance, in the first approach, the players table would include a "country\_name" attribute. Conversely, in the second approach, the players table would have a "country\_id" attribute. To retrieve the name of the country to which a player belongs, a JOIN operation is necessary, linking the players table with the countries table where tuples containing the country id and name are stored.

While the initial approach reduces the number of JOIN operations for each attribute, it may lead to slower query execution times when searching for items with long textual variables. This is because comparing an integer field is generally faster than comparing a varchar field. To determine the optimal approach, both methods were implemented and the query times for project queries were compared. Although both approaches yielded similar query times for most questions, the second approach (utilizing separate tables for such attributes) consistently demonstrated more stable query performance. As a result, it was selected the preferable approach.

The ER diagram for this project (page 5) has been saved as an SVG image, allowing you to zoom in to view the details. However, some texts in diamonds have moved out of the place. You can download the image from here:

<https://drive.google.com/drive/folders/1i5tU_tl41MoFQ2SjMl8M_ZlKO2fVUo3s?usp=sharing>

To make the visualization of event entities possible, a semi-entity[[1]](#footnote-1) consisting of shared attributes is created. This semi-entity, named "Shared Event Attributes", consists of attributes that should be present in all event entities. While a table has been created for this semi-entity, it does not correspond to any single entity in the original or final databases.



# Reduction to Relation Schemas

**countries**

sn, country\_id, country\_name

PK (sn)

**stadiums**

stadium\_id, stadium\_name

FK (country\_id), PK (stadium\_id)

**referees**

referee\_id, referee\_name

FK (country\_id), PK (referee\_id)

**managers**

manager\_id, manager\_name, manager\_nickname, manager\_dob,

FK (country\_id), PK (manager\_id)

**competitions\_seasons**

competition\_id, competition\_orig\_id, competition\_name, competition\_gender,

competition\_youth, competition\_international, country\_name, season\_id,

season\_name,

PK (competition\_id),

**teams**

team\_id,team\_orig\_id,team\_name,team\_gender,team\_group

FK (country\_id), FK (competition\_id), PK (team\_id),

**team\_managers**

FK (manager\_id), FK (team\_id), PK (manager\_id, team\_id)

**competition\_stages**

competition\_stage\_id, competition\_stage\_name,

PK (competition\_stage\_id)

**matches**

match\_id, match\_date, match\_kickoff, homeTeam\_score, awayTeam\_score, match\_week,

FK (homeTeam\_id), FK (awayTeam\_id), FK (stadium\_id), FK (referee\_id),

FK (competition\_id), FK (competition\_stage\_id), PK (match\_id),

**players**

player\_id, player\_name, player\_nickname,

FK (country\_id), PK (player\_id)

**player\_jerseys**

player\_id, jersey\_number,

FK (competition\_id), PK (player\_id, competition\_id)

**player\_positions**

player\_position\_id, player\_position\_name,

PK (player\_position\_id)

**player\_match\_positions**

time\_from, time\_to, from\_period, to\_period, start\_reason, end\_reason,

FK (match\_id), FK (team\_id), FK (player\_id), FK (player\_position\_id),

PK (match\_id, player\_id, player\_position\_id, time\_from)

**player\_match\_cards**

card\_type, time, reason, period,

FK (match\_id), FK (team\_id), FK (player\_id), PK (match\_id, player\_id, time)

**lineups**

FK (match\_id), FK (team\_id), FK (player\_id),

PK (match\_id, team\_id, player\_id)

**play\_patterns**

play\_pattern\_id, play\_pattern\_name,

PK (play\_pattern\_id)

**events**

event\_id, event\_db\_id, index, period, minute, second, location\_x, location\_y, event\_type\_name, possession\_count, duration, under\_pressure, counterpress,

FK (possession\_team\_id), FK (play\_pattern\_id), FK (player\_position\_id),

FK (match\_id), FK (team\_id), FK (player\_id), PK (event\_id)

**related\_events**

re\_id, event\_db\_id, related\_event\_db\_id,

PK (re\_id)

**event\_outcomes**

event\_outcome\_id, event\_outcome\_name,

PK (event\_outcome\_id)

**pass\_heights**

pass\_height\_id, pass\_height\_name,

PK (pass\_height\_id)

**body\_parts**

body\_part\_id, body\_part\_name,

PK (body\_part\_id)

**shot\_techniques**

shot\_technique\_id, shot\_technique\_name,

PK (shot\_technique\_id)

**pass\_types**

pass\_type\_id, pass\_type\_name,

PK (pass\_type\_id)

**foul\_types**

foul\_type\_id, foul\_type\_name,

PK (foul\_type\_id)

**ball\_receipts**

br\_id,

FK (event\_id), FK (event\_outcome\_id), PK (br\_id)

**ball\_carries**

bc\_id, end\_location\_x, end\_location\_y,

FK (event\_id), PK (bc\_id)

**events\_passes**

event\_id, event\_db\_id, index, period, minute, second, possession\_count,

location\_x, location\_y, duration, under\_pressure, counterpress, pass\_length,

pass\_angle, end\_location\_x, end\_location\_y, assisted\_shot\_id, backheel,

deflected, miscommunication, is\_cross, cut\_back, switch, shot\_assist,

goal\_assist,

FK (pass\_height\_id), FK (body\_part\_id), FK (recipient\_id), FK (match\_id),

FK (possession\_team\_id), FK (play\_pattern\_id), FK (player\_position\_id),

FK (team\_id), FK (player\_id), FK (pass\_type\_id), FK (event\_outcome\_id),

FK (pass\_technique\_id), PK (event\_pass\_id)

**ball\_recoveries**

br\_id, recovery\_failure, offensive,

FK (event\_id), PK (br\_id)

**duels**

duel\_id, duel\_type\_id, duel\_type\_name

FK (event\_id), FK (event\_outcome\_id), PK (duel\_id)

**blocks**

block\_id, deflection, offensive, save\_block,

FK (event\_id), PK (block\_id)

**clearances**

clearance\_id, aerial\_won,

FK (event\_id), FK (body\_part\_id), PK (clearance\_id)

**interceptions**

interception\_id,

FK (event\_id), FK (event\_outcome\_id), PK (interception\_id)

**dribbles**

dribble\_id, overrun, nutmeg, no\_touch,

FK (event\_id), FK (event\_outcome\_id), PK (dribble\_id)

**events\_shots**

event\_id, event\_db\_id, index, period, minute, second, possession\_count,

location\_x, location\_y, duration, under\_pressure, counterpress, key\_pass\_id,

aerial\_won, follows\_dribble, first\_time, open\_goal, deflected, statsbomb\_xg,

end\_location\_x, end\_location\_y, end\_location\_z,

FK (match\_id), FK (team\_id), FK (player\_id), FK (possession\_team\_id),

FK (play\_pattern\_id), FK (player\_position\_id), FK (body\_part\_id),

FK (technique\_id), FK (shot\_type\_id), FK (event\_outcome\_id), PK (event\_id)

**substitutions**

substitution\_id,

FK (event\_id), FK (event\_outcome\_id), FK (replacement\_id),

PK (substitution\_id)

**foul\_wons**

foul\_won\_id, defensive, advantage, penalty,

FK (event\_id), PK (foul\_won\_id)

**foul\_commits**

foul\_commit\_id, offensive, advantage, penalty, card\_type,

FK (event\_id), FK (foul\_type\_id), PK (foul\_commit\_id)

**goalkeeper\_actions**

gk\_action\_id, gk\_position, gk\_technique, gk\_action\_type,

FK (event\_id), FK (body\_part\_id), FK (event\_outcome\_id), PK (gk\_action\_id)

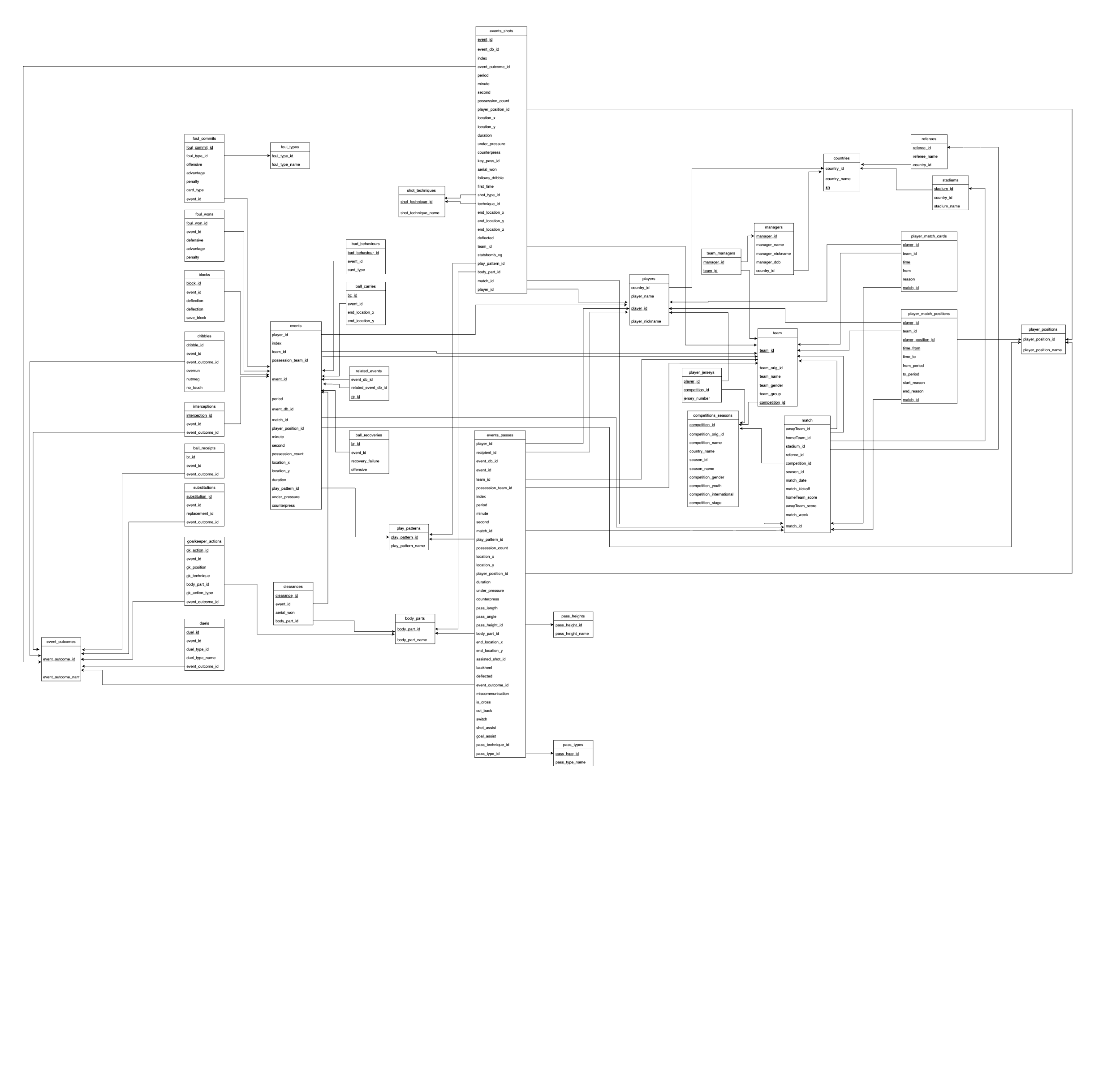
**bad\_behaviours**

bad\_behaviour\_id, card\_type,

FK (event\_id), PK (bad\_behaviour\_id)

# Database Schema Diagram

The diagram below has been saved as an SVG image, allowing you to zoom in to view the details. You can download the file from [Google Drive](https://drive.google.com/drive/folders/1i5tU_tl41MoFQ2SjMl8M_ZlKO2fVUo3s?usp=sharing).



# Bonus Queries

The solution to bonus queries are

1.

SELECT player\_name, COUNT(\*) as shot\_count

FROM competitions\_seasons

JOIN matches

ON competitions\_seasons.competition\_id = matches.competition\_id

JOIN events\_shots

ON matches.match\_id = events\_shots.match\_id

AND (end\_location\_x >= 119.9)

AND (end\_location\_z >= 2.67\*2/3 AND end\_location\_z <= 2.67)

AND ((end\_location\_y >= 36 AND end\_location\_y <= 36+8/3)

OR (end\_location\_y >= 44-8/3 AND end\_location\_y <= 44))

JOIN players

ON events\_shots.player\_id = players.player\_id

WHERE competitions\_seasons.competition\_name = 'La Liga'

AND (competitions\_seasons.season\_name = '2018/2019'

OR competitions\_seasons.season\_name = '2019/2020'

OR competitions\_seasons.season\_name = '2020/2021')

GROUP BY player\_name

ORDER BY shot\_count DESC;

2.

SELECT team\_name, COUNT(\*) as pass\_count

FROM competitions\_seasons

JOIN matches

ON competitions\_seasons.competition\_id = matches.competition\_id

AND competitions\_seasons.competition\_name = 'La Liga'

AND competitions\_seasons.season\_name = '2020/2021'

JOIN events\_passes

ON matches.match\_id = events\_passes.match\_id

AND (end\_location\_x >= 102.4 AND end\_location\_y >= 19.9 AND end\_location\_y <= 60.1)

JOIN teams

ON events\_passes.team\_id = teams.team\_id

GROUP BY team\_name

ORDER BY pass\_count DESC;

You can find the YouTube video of the results here: <https://youtu.be/WSV_4Svxqj8>

1. Semi-entity is not a technical term. It’s being used here to indicate an object with attributes like an entity but not being an actual entity. [↑](#footnote-ref-1)