DBM 1- Databases

Conceptualization of a database - From an ER model to the SQL implementation and queries in SQL and relational algebra

Authors: Brian O’Sullivan, Eike Stoltze

Domain: FIFA World Cup

This Database contains the data for every football match played in the final rounds of FIFA World Cups since the beginning in 1930.

Introduction

This report summarises the process of creating a database. This process is split into the following steps:

1. Finding a domain and data
2. Developing an ER-Model
3. Creating the logical schema
4. SQL implementation
5. Performing queries on the database with SQL commands

All steps will be explained in the report below and should give a good overview of the process of creating a database.

1. Finding a Domain and data

The domain of this project is the final rounds of FIFA World Cups. The FIFA World Cup is a tournament that is held every four years and organized by FIFA, the world's biggest football association. Many nations all around the globe qualify beforehand and play a set of matches to determine who's the best footballing nation. These matches are seen by huge parts of the world, which is why nearly all of them are well documented.

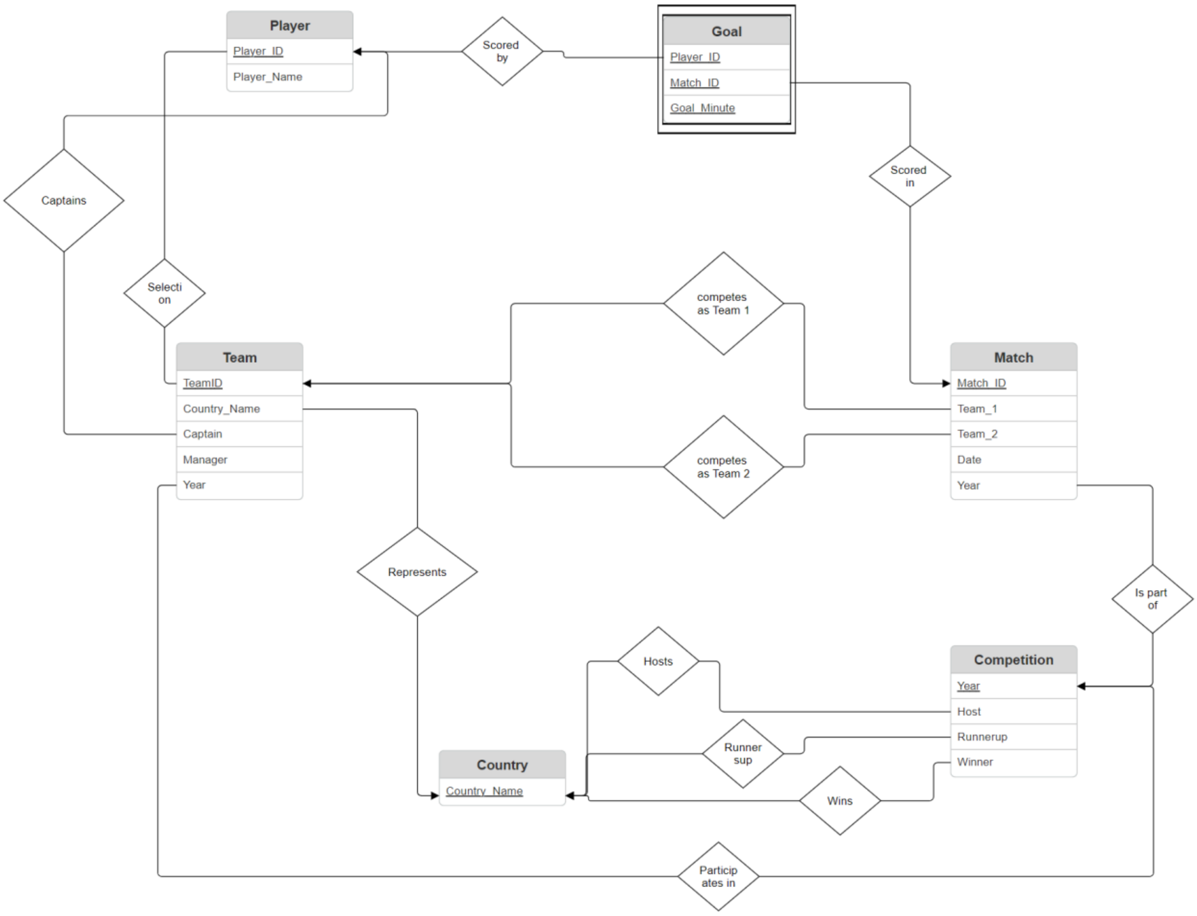
The data used in this project was taken from *Kaggle.com*, a website that gives access to a large amount of data for free. The dataset "Football - FIFA World Cup, 1930 – 2022" contains a lot of details about every single match played in the final rounds of a World Cup, such as Cards given to whom and in what minute, attendance of every match, and so on. That's why it was studied extensively and reduced to the most interesting information. The attributes of the database, that were kept are the following: Countries names, their coaches and captains, the date and year of the match and all the goals scored during each match (excluding penalty shootouts).

2. Developing an ER-Model

The chosen data was formed into entities and relationships that describe the entire Dataset. We decided to create the following entities: Competition, Match, Player, Country, Team and Goal. To render this model more precisely, certain constraints have to be determined:

- A Tournament is held every 4 years and is hosted by country. For each tournament, there is one winner and one runner-up.
- A Match consists of two Teams playing against each other on a certain date and in a certain tournament
- A team has a country it represents, a manager, that coaches the team for the duration of the tournament and a captain, who leads the team for the duration of the tournament.
- Goals can be scored in a match by a certain player at a certain minute of the game. To differentiate the goals of one player in a single match, there can only be one goal per minute by the same player

With these constraints and all entities and Relationships determined, the following ER model was created:



All entities except one are strong entities, as their primary key (underlined attribute) doesn't include a foreign key. The entity "Goal" contains *Player_ID* and *Match_ID* in its primary key, thus making it a weak entity. That is shown in the model by the second outline around the entity.

The first Normal Form is achieved, as each attribute is atomic. The second and third normal forms are also given, as all non-candidate attributes only depend on the primary key and no transitive dependencies (attributes depend on a non-candidate key) exist. 4NF is not achieved, as we do deal with multivalued dependencies. For example, "TEAM" does include the attributes *Coach*, *Year,* and *Country*. Since only one coach is coaching a single team per tournament, the combination of *Year* and *Country* implies *Coach*, just like *Year* and *Coach* imply *Country*. Both sets ({*Year, Country*} and {*Year, Coach*}) are no superkeys.

With the help of this model, all entities and relationships can be transferred into the logical schema.

3. Creating the logical schema

The entities with all their attributes are as follows:

- COMPETITION(Year, Host, Winner, Runnerup)
- COUNTRY(Country_Name)
- GOAL(Player_ID, Match_ID, Minute)
- MATCH(Match_ID, Team_1, Team_2, Date, Year)
- PLAYER(Player_ID, Player_Name)

- TEAM(Team_ID, Country_Name, Captain, Manager, Year)

The underlined attributes are the primary key. This key is the smallest collection of one or multiple attributes that are needed to differentiate all tuples within the entities. Since there is only a World Cup every four years, the primary key of "COMPETITION" is *Year*. "MATCH" could be differentiated by *Team_1, Team_2,* and *Date*, but for the sake of simplicity, the Attribute *Match_ID* was added and is used as the primary key. The entity "PLAYER" needs to have the attribute *Player_ID* as there could be multiple players that share the same name. The Attribute *Player_Name* could also be split into *First_Name* and *Last_Name*. This is not relevant to our database, which is why we left this design choice out. All entries in the entity "PLAYER" are taken from the captains and goal scorers of each match. The entity "TEAM" has the foreign key *Year* which would be enough to differentiate all the teams that represent each country for the different tournaments. But similar to "Match", the Attribute *Team_ID* was created to give the Model a cleaner appearance. In the case of "GOAL", all attributes are needed to form a primary key. As said previously, the attribute *Goal_Minute* was created to differentiate multiple goals scored by one player in one match.

Regarding the mapping cardinality, nearly all relationships are "Many-to-One" relationships. Therefore, most relationships only result in an attribute becoming a foreign key within another relation/entity, instead of forming a new table. The only "Many-to-Many" relationship is:

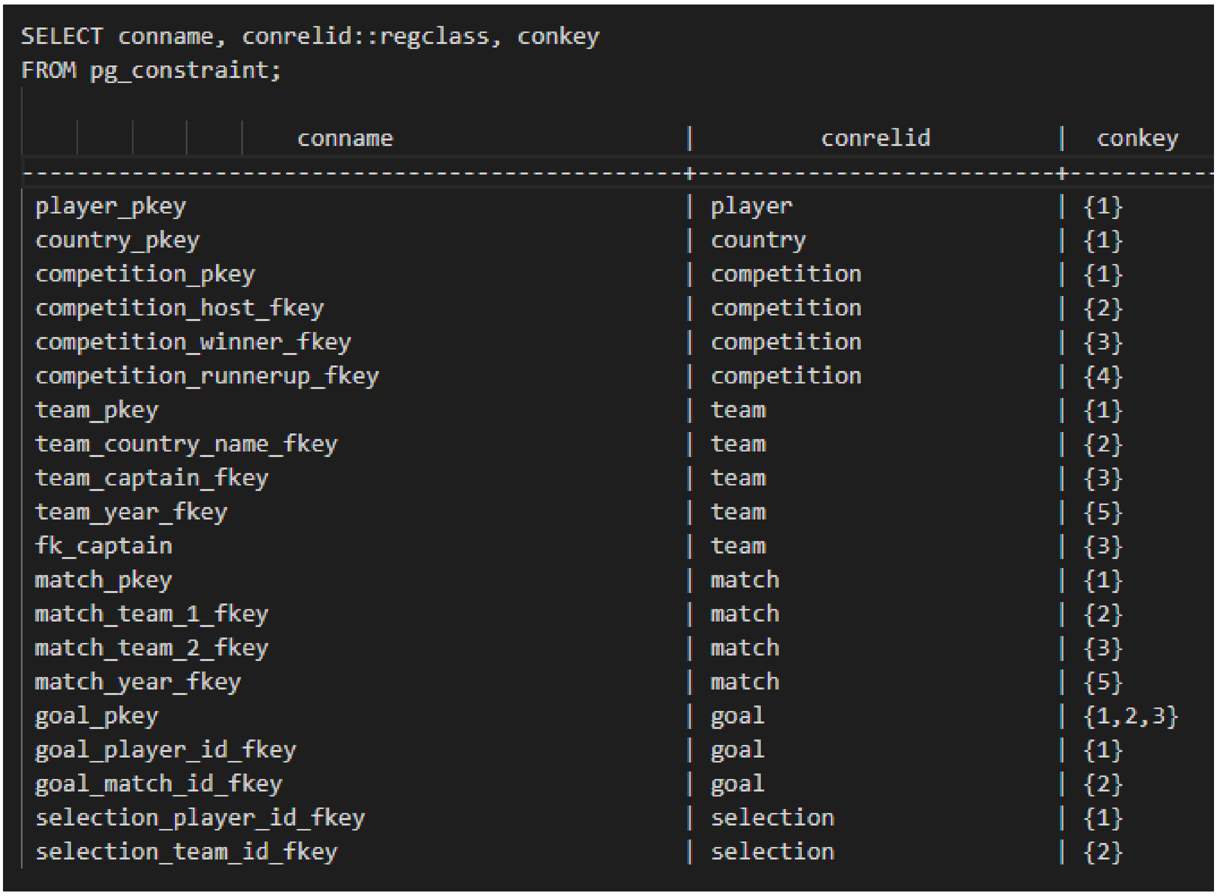
- SELECTION(Player_ID, Team_ID)

This relationship lists all the players, that either lead their team as captain or scored a goal. A team can therefore have multiple players, while a single player can also play in multiple teams. It is also possible for players to switch to another nationality, although this is now allowed by FIFA regulations these days.

All relationships are binary, which is desired.

4. SQL implementation/Problems with the dataset

Since the logical schema was already set up, creating the database and tables was an easy task, as all attributes were given in the logical schema. All primary and foreign keys must be declared.



```
SELECT conname, conrelid::regclass, conkey
FROM pg_constraint;

                  conname                   |         conrelid          | conkey
--------------------------------------------+---------------------------+----------
 player_pkey                                | player                    | {1}
 country_pkey                               | country                   | {1}
 competition_pkey                           | competition               | {1}
 competition_host_fkey                      | competition               | {2}
 competition_winner_fkey                    | competition               | {3}
 competition_runnerup_fkey                  | competition               | {4}
 team_pkey                                  | team                      | {1}
 team_country_name_fkey                     | team                      | {2}
 team_captain_fkey                          | team                      | {3}
 team_year_fkey                             | team                      | {5}
 fk_captain                                 | team                      | {3}
 match_pkey                                 | match                     | {1}
 match_team_1_fkey                          | match                     | {2}
 match_team_2_fkey                          | match                     | {3}
 match_year_fkey                            | match                     | {5}
 goal_pkey                                  | goal                      | {1,2,3}
 goal_player_id_fkey                        | goal                      | {1}
 goal_match_id_fkey                         | goal                      | {2}
 selection_player_id_fkey                   | selection                 | {1}
 selection_team_id_fkey                     | selection                 | {2}
```

To ensure the integrity of the database, certain constraints are put in place for each relation:

1. Competition Integrity:

- Must have a unique year,
- *Host*, *Winner,* and *Runnerup* must all be valid foreign keys referencing the COUNTRY table

2. Country Integrity:

- *Country_Name* must be unique (<255 chars)

3. Player Integrity:

- *Player_ID* must be unique
- *Player_Name* must be <255 chars

4. Team Integrity:

- Team_ID must be unique
- Country_name must reference a valid entry in the country table
- Captain must reference a valid Player_ID in the PLAYER table
- Manager must be <255 chars
- Year must reference a competition year

5. Selection Integrity:

- Team_id must reference a valid team_id in the team table
- Player_id must reference a valid player_id in the player table

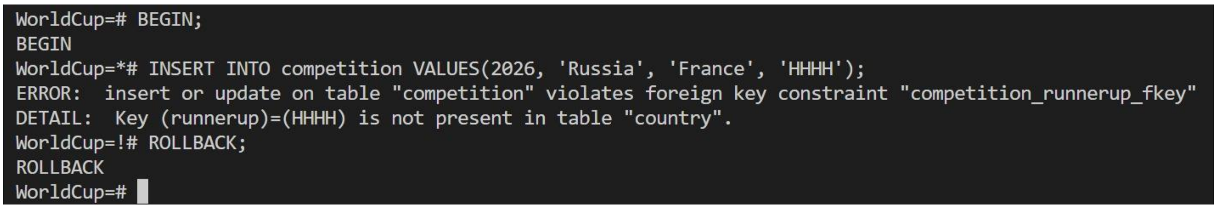
6. Match Integrity:

- Match_id must be unique
- Team_1 must reference a valid Team_id in the team table
- Team_2 must reference a valid Team_id in the team table
- Date must be no longer than 10 chars
- Year must reference a valid competition year in the competition table

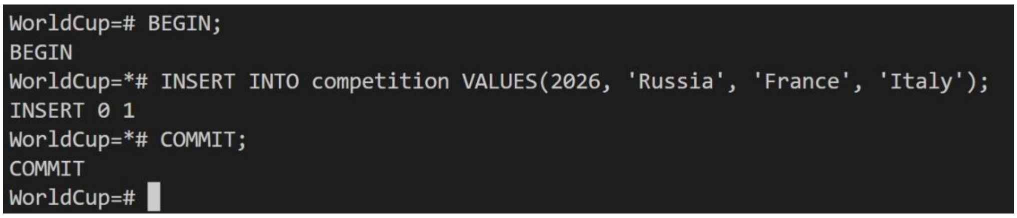
7. Goal Integrity:

- Player_id must reference a valid player_id in the player table
- Match_id must reference a valid match_id in the match table
- Minute must be less than 10 chars

We made responsible use of transaction control throughout the building of our database to ensure consistency and integrity. We took care when inserting, deleting, creating, and dropping tables by committing or rolling back based on the success of each statement. Examples below:



```
WorldCup=# BEGIN;
BEGIN
WorldCup=*# INSERT INTO competition VALUES(2026, 'Russia', 'France', 'HHHH');
ERROR:  insert or update on table "competition" violates foreign key constraint "competition_runnerup_fkey"
DETAIL:  Key (runnerup)=(HHHH) is not present in table "country".
WorldCup=!# ROLLBACK;
ROLLBACK
WorldCup=#
```



```
WorldCup=# BEGIN;
BEGIN
WorldCup=*# INSERT INTO competition VALUES(2026, 'Russia', 'France', 'Italy');
INSERT 0 1
WorldCup=*# COMMIT;
COMMIT
WorldCup=#
```

After the creation of all tables, the data was imported into the database. The first problems occurred when some string attributes appeared twice or caused trouble due to their content. In some instances, the seemingly same string (eg. “Lionel Messi”) was listed twice. That happened due to a different syntax for the space sign in between the first and last names. This error can be fixed by replacing “\xa0” with a regular space sign using the following line:

```
new_row = new_row.replace("\xa0", " ")
```

Similar mistakes happened with letters and names that include an apostrophe “ ’ ”, such as “Côte d'Ivoire”.

Another content-related mistake occurred, due to a change in countries' names. Some countries such as “Yugoslavia” were dissolved, so they can just be left out. But other countries changed their names or merges, such as the case with “Germany”. Both “East-” and “West Germany” played in at least one World Cup. Since their reunion in 1991, a new country with the name “Germany” was put in place in FIFA matches. Since “East Germany” never won an international competition, all of “West Germany” 's achievements also count as achievements of “Germany”. That’s why all entries of “West Germany” are replaced with “Germany” using a similar command as before:

```
    new_row = new_row.replace("West Germany", "Germany")
```

The World Cup in 2002 also caused trouble, as it was the first and, so far, only competition hosted by two nations. For the reason of simplicity, only “Japan” was chosen to be the assigned host for this World Cup. It could be solved by implementing a “Many-to-Many” relationship, in which all the Years of the competition are listed with all the hosts. This might be a better solution for future competitions, as the next two world cups (2026, 2030) will be held in multiple countries.

Attributing goals to specific players from the dataset was difficult as the goals were listed in string format e.g. “Ángel Di María · 36|Lionel Messi · 108|Kylian Mbappé · 81|Lionel Messi (P) · 23|Kylian Mbappé (P) · 80|Kylian Mbappé (P) · 118”. Therefore, we had to parse each goal to extract the minute and the player. Then we had to find the named player in our list of players to make sure our database was correctly synchronized.

Synchronizing our data was also difficult to get right. E.g. when creating our GOAL table each goal had to have a *Match_ID* and a *Player_ID*, so for each goal in each match our Python script had to search through our player's table and find the correct player’s name so that we could attribute the correct *Player_ID* to the goal.

We also came across missing data. Multiple captains and coaches from the squads of the tournaments held between 1998 and 2014 were missing. All these names had to be looked up and filled in manually. The data was taken from the Wikipedia articles of each squad.

5. Perform queries on the database with SQL commands

Before the creation of the ER model, a few Questions were asked in natural language. These are just questions that were formulated out of curiosity and that should possibly be answered within the database using SQL commands.

The following set of queries were asked:

1. Name all registered counties that start with the letter “C”.
2. How many times has the most successful country won the World Cup?
3. List all teams Italy sent to the World Cup (year, captain, and manager)
4. List all the games in which Kylian Mbappe scored.
5. List the 10 best goal scorers of Portugal.
6. Name the players, their countries, and the minute they scored in the 2022 final.

The queries were first formed into relational algebra expressions to manually form the asked relations. With the help of those, SQL commands can be performed. The RA and SQL commands with the corresponding results are listed and explained below.

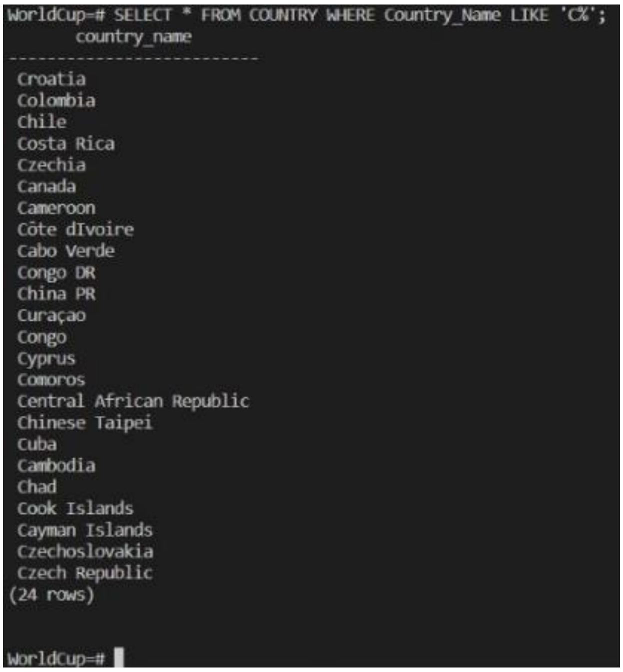
1. Name all registered counties that start with the letter “C”.

For this query, we simply select all countries, where the attribute *Country_Name* starts with the letter ”C”. The projection with the specification on attribute “*Country_Name*” is not necessary, as it is the only attribute of the entity COUNTRY.

$$\Pi_{Country\_Name}(\sigma_{County\_Name\,=\,"C\%"}(COUNTRY))$$

The translation to SQL is easy as well:

```
SELECT * FROM COUNTRY WHERE Country_Name LIKE 'C%';
```



```
WorldCup=# SELECT * FROM COUNTRY WHERE Country_Name LIKE 'C%';
       country_name
--------------------------
 Croatia
 Colombia
 Chile
 Costa Rica
 Czechia
 Canada
 Cameroon
 Côte dIvoire
 Cabo Verde
 Congo DR
 China PR
 Curaçao
 Congo
 Cyprus
 Comoros
 Central African Republic
 Chinese Taipei
 Cuba
 Cambodia
 Chad
 Cook Islands
 Cayman Islands
 Czechoslovakia
 Czech Republic
(24 rows)


WorldCup=#
```

2. How many times has the most successful country won the World Cup?

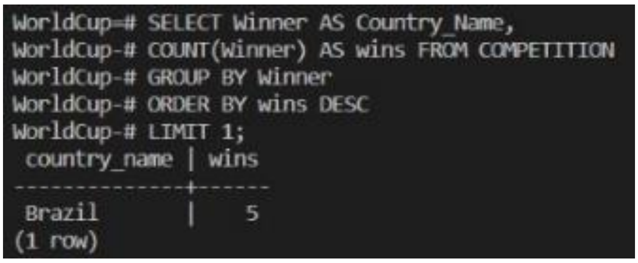
This query is not possible in relational algebra, as there are no “LIMIT” and “ORDER” operators. Only a list of winners can be chosen with the following command:

$$\Pi_{Winner}(COMPETITION)$$

Luckily the two needed operators are easy to implement in SQL:

```
SELECT Winner AS Country_Name,
COUNT(Winner) AS wins FROM COMPETITION
GROUP BY Winner
ORDER BY wins DESC
LIMIT 1;
```

This command selects all the tuples of the attribute *Winner* of the relation COMPETITION and renames these to “Country_Name”. In the next step, the number of occurrences of each Winner gets counted; each country gets assigned the corresponding number of wins. This table is then ordered by the number of wins in descending order and limited to the first entry from the top.



```
WorldCup=# SELECT Winner AS Country_Name,
WorldCup-# COUNT(Winner) AS wins FROM COMPETITION
WorldCup-# GROUP BY Winner
WorldCup-# ORDER BY wins DESC
WorldCup-# LIMIT 1;
 country_name | wins
--------------+------
 Brazil       |    5
(1 row)
```

3. List all teams Italy sent to the World Cup (year, captain, and manager)

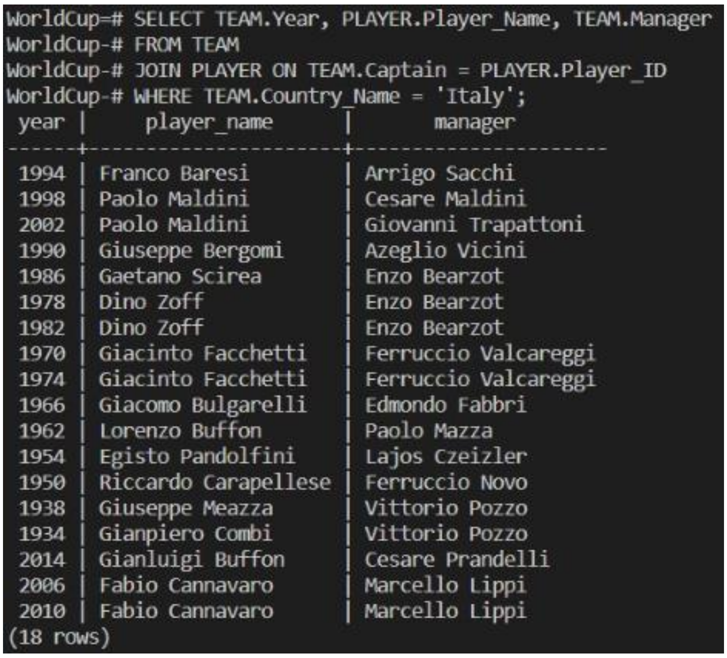
This query combines the Project, Join, and Select operators. We first choose all teams, where the attribute *Country_Name* is “Italy”. We then join this table with the table PLAYER on the *Player_ID* to get the names of the captains. Since we are only interested in the year, the captain's name, and the manager, we choose these attributes in our projection.

$$\Pi_X(\sigma_{County\_Name\,=\,"Italy"}(TEAM) \bowtie_{TEAM.Captain=PLAYER.Player\_ID} PLAYER)$$

$$\text{With } X = TEAM.Year, TEAM.Manager, PLAYER.Player\_Name$$

The SQL is easily formed following the structure of the RA.

```
SELECT TEAM.Year, PLAYER.Player_Name, TEAM.Manager

FROM TEAM

JOIN PLAYER ON TEAM.Captain = PLAYER.Player_ID

WHERE TEAM.Country_Name = 'Italy';
```



```
WorldCup=# SELECT TEAM.Year, PLAYER.Player_Name, TEAM.Manager
WorldCup-# FROM TEAM
WorldCup-# JOIN PLAYER ON TEAM.Captain = PLAYER.Player_ID
WorldCup-# WHERE TEAM.Country_Name = 'Italy';
 year |     player_name      |       manager
------+----------------------+---------------------
 1994 | Franco Baresi        | Arrigo Sacchi
 1998 | Paolo Maldini        | Cesare Maldini
 2002 | Paolo Maldini        | Giovanni Trapattoni
 1990 | Giuseppe Bergomi     | Azeglio Vicini
 1986 | Gaetano Scirea       | Enzo Bearzot
 1978 | Dino Zoff            | Enzo Bearzot
 1982 | Dino Zoff            | Enzo Bearzot
 1970 | Giacinto Facchetti   | Ferruccio Valcareggi
 1974 | Giacinto Facchetti   | Ferruccio Valcareggi
 1966 | Giacomo Bulgarelli   | Edmondo Fabbri
 1962 | Lorenzo Buffon       | Paolo Mazza
 1954 | Egisto Pandolfini    | Lajos Czeizler
 1950 | Riccardo Carapellese | Ferruccio Novo
 1938 | Giuseppe Meazza      | Vittorio Pozzo
 1934 | Gianpiero Combi      | Vittorio Pozzo
 2014 | Gianluigi Buffon     | Cesare Prandelli
 2006 | Fabio Cannavaro      | Marcello Lippi
 2010 | Fabio Cannavaro      | Marcello Lippi
(18 rows)
```

4. List all the games in which Kylian Mbappe scored.

This query is similar to the last one, as it is also a combination of Project, Select, and Join. In this case, we choose the player's name from the table PLAYER and join that with the relation GOAL on the base of *Player_ID* to combine the goals with the name. After that, we project *Match_ID*, *Player_Name,* and *Minute*.

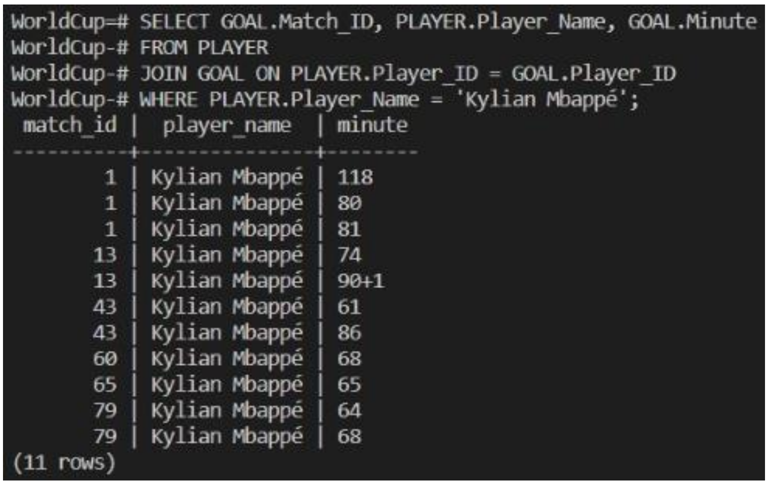
$$\Pi_{Match\_ID,Player\_Name,Minute}(\sigma_{Player\_Name\,=\,"Kylian\ Mbappe"}(PLAYER)\bowtie_{Player\_ID} GOAL)$$

This will also result in a similar structure as the SQL command from query 3.

SELECT GOAL.Match_ID, PLAYER.Player_Name, GOAL.Minute FROM PLAYER

JOIN GOAL ON PLAYER.Player_ID = GOAL.Player_ID

WHERE PLAYER.Player_Name = 'Kylian Mbappé';



```
WorldCup=# SELECT GOAL.Match_ID, PLAYER.Player_Name, GOAL.Minute
WorldCup-# FROM PLAYER
WorldCup-# JOIN GOAL ON PLAYER.Player_ID = GOAL.Player_ID
WorldCup-# WHERE PLAYER.Player_Name = 'Kylian Mbappé';
 match_id |  player_name  | minute
----------+---------------+--------
        1 | Kylian Mbappé | 118
        1 | Kylian Mbappé | 80
        1 | Kylian Mbappé | 81
       13 | Kylian Mbappé | 74
       13 | Kylian Mbappé | 90+1
       43 | Kylian Mbappé | 61
       43 | Kylian Mbappé | 86
       60 | Kylian Mbappé | 68
       65 | Kylian Mbappé | 65
       79 | Kylian Mbappé | 64
       79 | Kylian Mbappé | 68
(11 rows)
```

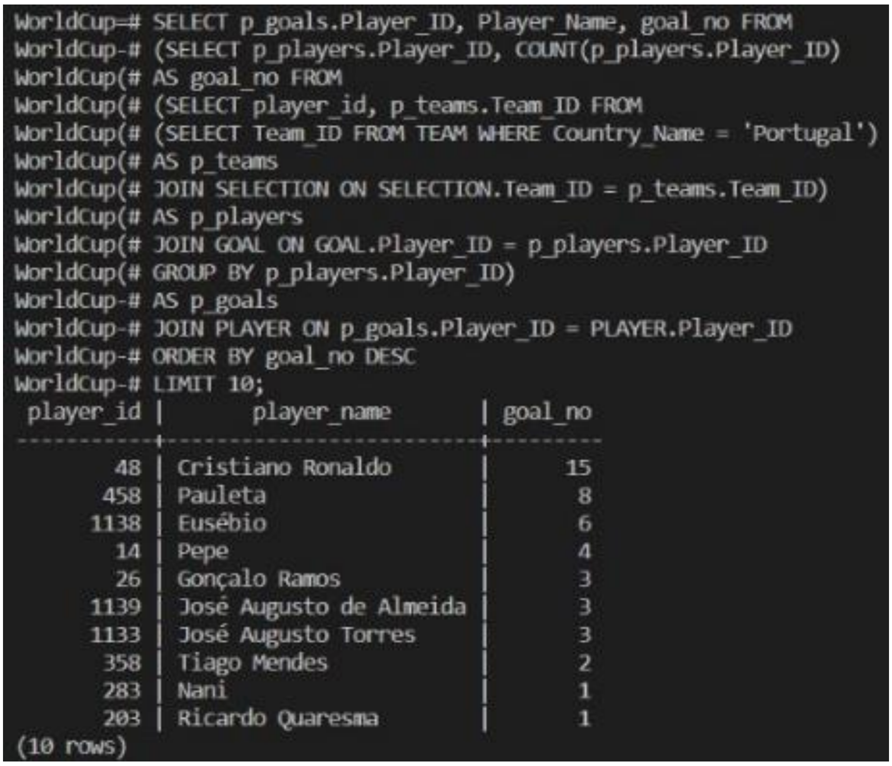
5. List the 10 best goal scorers of Portugal.

This query has similar issues to Nr. 2, as we can neither order the table by a certain attribute nor count the number of entries. Instead, we only choose to name all tuples with the *Match_ID*s and the players' names who scored in these games.

$$\Pi_{PLAYER.Player\_Name,GOAL.Match\_ID}(((\sigma_{Country\_Name\,=\,``Portugal"}(COUNTRY) \bowtie_{Team\_ID} SELECTION) \bowtie_{Player\_ID} GOAL) \bowtie_{Player\_ID} PLAYER)$$

The SQL command is slightly more complex, as we want to show the player's ID and name and number of goals they scored. We look up all players (p_players) who played for Portuguese teams (p_teams) by joining SELECTION and TEAM. We can join this on GOAL to find all the goals scored by players who played for Portuguese teams. By counting the number of entries of each *Player_ID* we can extract the number of goals they scored. By selecting our chosen attributes, limiting the table to 10 tuples, and putting them in a descending order, we get our desired table.

```
SELECT p_goals.Player_ID, Player_Name, goal_no FROM

(SELECT p_players.Player_ID, COUNT(p_players.Player_ID)

AS goal_no FROM

(SELECT player_id, p_teams.Team_ID FROM

(SELECT Team_ID FROM TEAM WHERE Country_Name = 'Portugal')

AS p_teams

JOIN SELECTION ON SELECTION.Team_ID = p_teams.Team_ID)

AS p_players

JOIN GOAL ON GOAL.Player_ID = p_players.Player_ID

GROUP BY p_players.Player_ID)

AS p_goals

JOIN PLAYER ON p_goals.Player_ID = PLAYER.Player_ID

ORDER BY goal_no DESC

LIMIT 10;
```



```
WorldCup=# SELECT p_goals.Player_ID, Player_Name, goal_no FROM
WorldCup-# (SELECT p_players.Player_ID, COUNT(p_players.Player_ID)
WorldCup(# AS goal_no FROM
WorldCup(# (SELECT player_id, p_teams.Team_ID FROM
WorldCup(# (SELECT Team_ID FROM TEAM WHERE Country_Name = 'Portugal')
WorldCup(# AS p_teams
WorldCup(# JOIN SELECTION ON SELECTION.Team_ID = p_teams.Team_ID)
WorldCup(# AS p_players
WorldCup(# JOIN GOAL ON GOAL.Player_ID = p_players.Player_ID
WorldCup(# GROUP BY p_players.Player_ID)
WorldCup-# AS p_goals
WorldCup-# JOIN PLAYER ON p_goals.Player_ID = PLAYER.Player_ID
WorldCup-# ORDER BY goal_no DESC
WorldCup-# LIMIT 10;
 player_id |       player_name        | goal_no
-----------+--------------------------+---------
        48 | Cristiano Ronaldo        |      15
       458 | Pauleta                  |       8
      1138 | Eusébio                  |       6
        14 | Pepe                     |       4
        26 | Gonçalo Ramos            |       3
      1139 | José Augusto de Almeida  |       3
      1133 | José Augusto Torres      |       3
       358 | Tiago Mendes             |       2
       283 | Nani                     |       1
       203 | Ricardo Quaresma         |       1
(10 rows)
```

6. Name the players, their countries, and the minute they scored in the 2022 final.

For this Query, we need to either look up the Date of the match and use this as the predicate in our selection operator or we can manually look up the *Match_ID* and apply it. We join the selected MATCH and GOAL on *Match_ID* to end up with all the goals scored in our chosen match. We can join this with PLAYER and SELECTION on *Player_ID*. Another Join operation needs to be done to specify the players' country. We select the from TEAM where TEAM.*Year*=MATCH.*Year* to only get the teams of the Year the match took place. This table shows us all goals scored in the chosen match (final 2022), the goal scorers, and their team info. By projecting *Team_Country, Player_Name,* and *Minute* we get our desired table.

$$\Pi_{Team\_Country,Player\_Name,Minute}\left(\begin{array}{c}\Big(\Big(\big(\sigma_{Match\_ID\,=\,"1"}(MATCH)\bowtie_{Match\_ID} GOAL\big)\bowtie_{Player\_ID} PLAYER\Big)\\ \bowtie_{Player\_ID} SELECTION\Big)\bowtie_{Year}\sigma_{TEAM.Year=MATCH.Year}(TEAM)\end{array}\right)$$

We can easily transfer this command to SQL:

SELECT Country_Name, Player_Name, Minute FROM

(SELECT Team_ID, Player_Name, Minute FROM

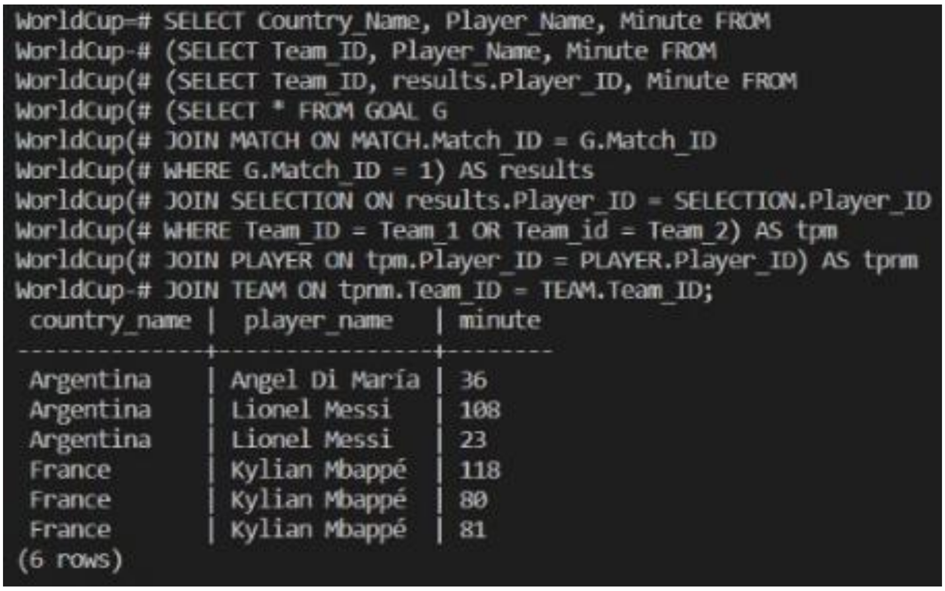
(SELECT Team_ID, results.Player_ID, Minute FROM

(SELECT * FROM GOAL G

JOIN MATCH ON MATCH.Match_ID = G.Match_ID

WHERE G.Match_ID = 1) AS results

```
JOIN SELECTION ON results.Player_ID = SELECTION.Player_ID
WHERE Team_ID = Team_1 OR Team_id = Team_2) AS tpm
JOIN PLAYER ON tpm.Player_ID = PLAYER.Player_ID) AS tpnm
JOIN TEAM ON tpnm.Team_ID = TEAM.Team_ID;
```



```
WorldCup=# SELECT Country_Name, Player_Name, Minute FROM
WorldCup-# (SELECT Team_ID, Player_Name, Minute FROM
WorldCup(# (SELECT Team_ID, results.Player_ID, Minute FROM
WorldCup(# (SELECT * FROM GOAL G
WorldCup(# JOIN MATCH ON MATCH.Match_ID = G.Match_ID
WorldCup(# WHERE G.Match_ID = 1) AS results
WorldCup(# JOIN SELECTION ON results.Player_ID = SELECTION.Player_ID
WorldCup(# WHERE Team_ID = Team_1 OR Team_id = Team_2) AS tpm
WorldCup(# JOIN PLAYER ON tpm.Player_ID = PLAYER.Player_ID) AS tpnm
WorldCup-# JOIN TEAM ON tpnm.Team_ID = TEAM.Team_ID;
 country_name |  player_name   | minute
--------------+----------------+--------
 Argentina    | Angel Di María | 36
 Argentina    | Lionel Messi   | 108
 Argentina    | Lionel Messi   | 23
 France       | Kylian Mbappé  | 118
 France       | Kylian Mbappé  | 80
 France       | Kylian Mbappé  | 81
(6 rows)
```

Summary

The created database is a solid foundation to store the match fixtures, goals, players, squads, and competitions. All relationships are binary and provide a great way of storing data and retrieving information. The structure allows a variety of queries to be asked.

Issues might occur when future competitions will be added, as a multitude of hosts is not foreseen in this model. This however can be fixed by implementing a new "Many-to-Many" relationship HOST between COUNTRY and COMPETITION and removing the attribute *Host* from COMPETITION. Also, 4NF can be achieved by changing the structure of the entity "TEAM" to avoid a multivalued dependency between the attributes *Year*, *Coach,* and *Country*.