Databases (L.EIC012)

Creating and Querying a Database

UEFA Champions League Results Management

Authors:
Davide Teixeira (up202109860)
Inês Aidos (up201107323)
Teresa Moreira (up201607371)

Professors:
Michel Celestino Paiva Ferreira
Gabriel de Sousa Torcato David
Daniel Luís Gonçalves Garrido
Pedro Emanuel Cardoso de Sousa
Ahmad Naser Eddin
Lázaro Gabriel Barros da Costa
Pedro Gonçalo Ferreira Alves Nogueira
António Humberto Sá Pinto

Bachelor in Informatics and Computing Engineering
Faculdade de Engenharia da Universidade do Porto

2022/2023

Work delivered on: 21/12/2022

Introduction

This project aims to create and interrogate a database to manage the results of the UEFA Champions League. This database must respond to the release of results day by day, goal scorers, teams that play at home and play away. It is intended to manage the group stage and the knockout stage until the final where the winner is found. It is intended to manage the financial amount that each team raised in the competition.

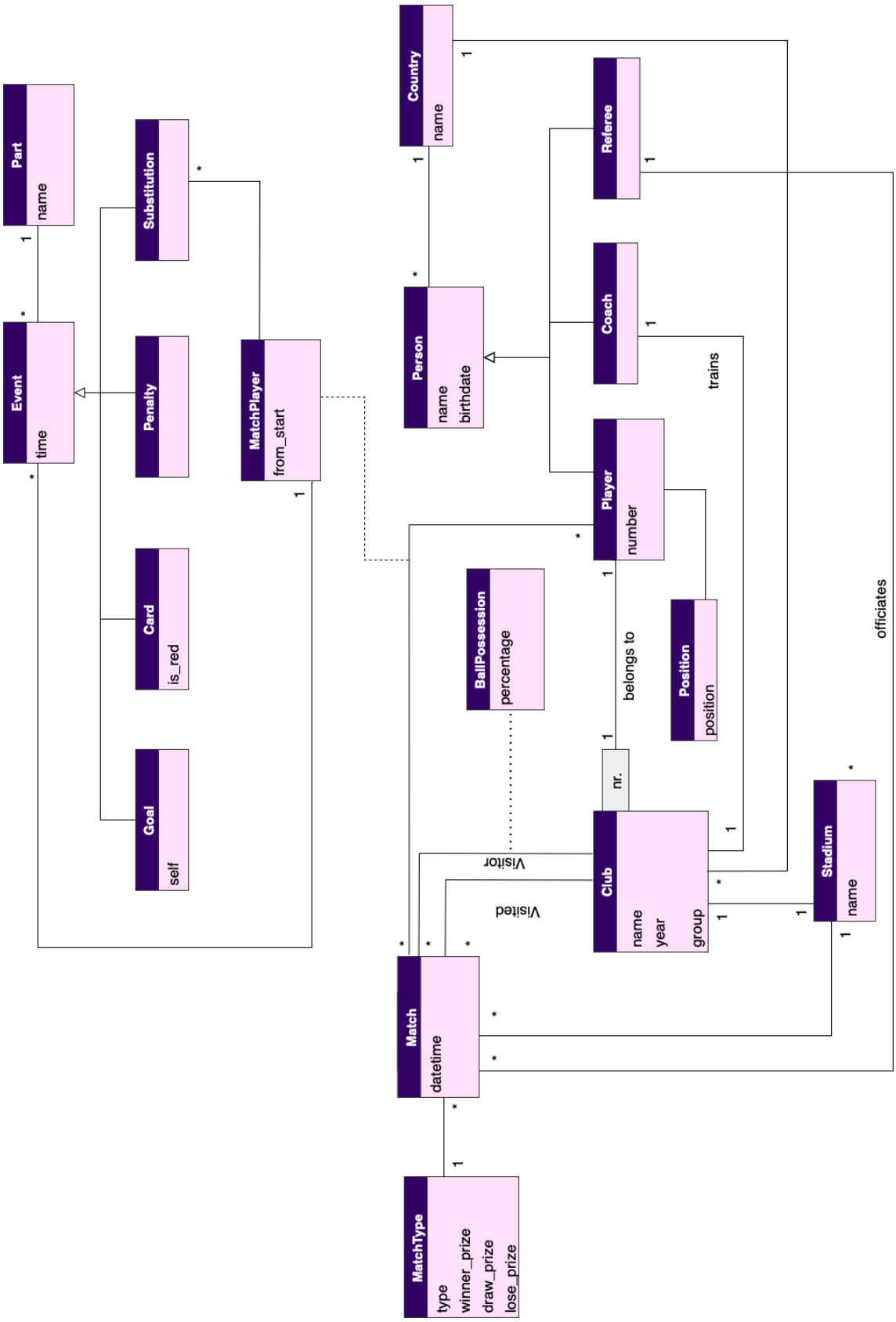
To this end, a conceptual model was developed for the topic in question, this model was mapped to a relational schema, the schema was implemented in an SQLite database, data was introduced and, finally, the database was queried.

Notes

This new delivery has some differences compared to the first delivery in tasks that were requested in the first delivery:

- We changed the “Pragma Foreign Keys” to “Pragma Foreign Keys = ON” and solved the errors that used to appear if we changed that.
- We changed the attribute “groupp” to group
- Some Camel Case attributes were changed to Snake Case, as requested
- More restrictions were added to numeric attributes ensuring they are greater than zero

UML Diagram


Part
name
Substitution
Country
name
Referee
Event
time
Penalty
MatchPlayer
from_start
Person
name
birthdate
Coach
trains
Player
number
BallPossession
percentage
belongs to
Position
position
Card
is_red
Goal
self
nr.
Club
name
year
group
Stadium
name
officiates
Visitor
Visited
Match
datetime
MatchType
type
winner_prize
draw_prize
lose_prize

Relational Schemas Definition

Club (idClub, name, year, group, idStadium->Stadium, idCountry->Country, idCoach->Coach);

Position (idPosition, position);

Person (idPerson, name, birthdate, idCountry->Country);

Player (idPlayer->Person, number, idPosition->Position , idClub ->Club);

Referee (idReferee->Person);

Coach (idCoach->Person);

Country (idCountry, name);

Match (idMatch, datetime, idVisited->Club, idVisitor->Club, idStadium->Stadium, idMatchType->MatchType, idReferee->Referee);

MatchType(idMatchType, type, winner_prize, draw_prize, lose_prize);

MatchPlayer(idMatch->Match, idPlayer->Player, from_start);

Event(idEvent, {idMatch, idPlayer}->MatchPlayer, time, idPart->Part);

Part(idPart, name);

Goal(idGoal->Event, self);

Card(idCard->Event, is_red);

Penalty(idPenalty->Event);

Substitution(idSubstitution->Event, {idMatch, idPlayer}->MatchPlayer);

BallPossession (idMatch->Match, idClub->Club, percentage);

Stadium (idStadium, name);

Functional Dependencies and Normal Forms Analysis

There are two main justifications for a relation to be in the BCNF(Boyce-Codd Normal Form) and, therefore, in the 3FN

- A relation R is in BCNF if and only if: whenever there is a nontrivial FD A1 A2 ...An ->B1 B2 ...Bm for R, it is the case that {A1,A2,...,An} is a superkey for R.
- The relation R does not have nontrivial FD

- Club (idClub, name, year, group, idStadium->Stadium, idCountry->Country, idCoach->Coach);

FD: idClub -> name, year, idStadium, idCountry, idCoach
BCNF/3NF ✅ - First Justification

- Position(idPosition, position);

FD: IdPosition -> position
BCNF/3NF ✅ - First Justification

- Person (idPerson, name, birthdate, idCountry->Country);

FD: idPerson -> name, birthdate, idCountry
BCNF/3NF ✅ - First Justification

- Player (idPlayer->Person, number, idPosition->Position , idClub ->Club);

FD: idPlayer -> number, idPosition, idClub; number, idClub -> idPlayer, idPosition;
BCNF/3NF ✅ - First Justification

- Referee (idReferee->Person)
  BCNF/3NF ✅ - Second Justification

- Coach (idCoach->Person)
  BCNF/3NF ✅ - Second Justification

- Country (idCountry, name)

FD: idCountry -> name
BCNF/3NF ✅ - First Justification

- Match (idMatch, datetime, idVisited->Club, idVisitor->Club, idStadium->Stadium,, idMatchType->MatchType, idReferee->Referee);

FD: idMatch -> datetime, idVisited, idVisitor, idStadium, idMatchType, idReferee
BCNF/3NF ✅ - First Justification

- MatchType(idMatchType, type, winner_prize, draw_prize, lose_prize);

FD: idMatchType -> type, winner_prize, draw_prize, lose_prize
BCNF/3NF ✅ - First Justification

- MatchPlayer(idMatch->Match, idPlayer->Player, from_start);

FD: idMatch, idPlayer -> from_start

BCNF/3NF ☑ - First Justification

- Event(idEvent, {idMatch, idPlayer}->MatchPlayer, time, idPart->Part);

FD: idEvent -> idMatch, idPlayer, time, idPart

BCNF/3NF ☑ - First Justification

- Part(idPart, name);

FD: idPart -> name

BCNF/3NF ☑ - First Justification

- Goal (idGoal->Event, self)

FD: idGoal -> self

BCNF/3NF ☑ - First Justification

- Card(idCard->Event, is_red)

FD: idCard -> is_red

BCNF/3NF ☑ - First Justification

- Penalty(idPenalty->Event)

BCNF/3NF ☑ - Second Justification

- Substitution(idSubstitution->Event, {idMatch, idPlayer}->MatchPlayer);

FD: idSubstitution ->idMatch, idPlayer

BCNF/3NF ☑ - First Justification

- BallPossession (idMatch->Match, idClub->Club, percentage);

FD: idMatch, idClub -> percentage

BCNF/3NF ☑ - First Justification

- Stadium (idStadium, name)

FD: idStadium -> name

BCNF/3NF ☑ - First Justification

List and Form of Implementation of Constraints

In this work we used some constraints to avoid redundancy and to make the database coherent and secure.

Constraints Index:

- **NOT NULL:** The attribute (or tuple of attributes) must have a value and cannot be "Null".
- **DEFAULT:** In the case of inexistence of the attribute, it is assigned a default attribute
- **CHECK:** Checks if the attribute obeys a certain condition.
- **UNIQUE:** The attribute (or tuple of attributes) must be unique on that relation.
- **PRIMARY KEY:** The attribute (or tuple of attributes) defines each of the objects of the same class. It also adds the constraints "**NOT NULL**" and "**UNIQUE**".
- **FOREIGN KEY:** The attribute (or tuple of attributes) defines another object, in the same class or not.

Club

- idClub **PRIMARY KEY**
- name **NOT NULL**
- year **CHECK (year <= 2022)**
- idStadim **FOREIGN KEY Country**
- idCoach **FOREIGN Key Coach**

Position

- idPosition **PRIMARY KEY**
- position **NOT NULL**

Person

- idPerson **PRIMARY KEY**
- idName **NOT NULL**
- idCountry **FOREIGN KEY Country**

Player

- idPlayer **PRIMARY KEY; FOREIGN KEY Person**
- number **CHECK (0 < number < 100)**
- idPosition **FOREIGN KEY Position**
- idClub **FOREIGN KEY Club**

Referee

- idReferee **PRIMARY KEY; FOREIGN KEY Person**

Coach

- idCoach **PRIMARY KEY; FOREIGN KEY Person**

Country

- idCountry **PRIMARY KEY**
- name **NOT NULL**

Match
- idMatch **PRIMARY KEY**
- datetime **NOT NULL**
- idVisited **FOREIGN KEY Club**
- idVisitor **FOREIGN KEY Club**
- idStadium **FOREIGN KEY Stadium**
- idMatchType **FOREIGN KEY MatchType**
- idReferee **FOREIGN KEY Referee**

MatchType
- idType **PRIMARY KEY**
- type **NOT NULL**
- winner_Prize **NOT NULL CHECK (winner_prize > 0)**
- draw_Prize **CHECK (draw_prize >= 0)**
- lose_Prize **CHECK (lose_prize >= 0)**

MatchPlayer
- {idMatch, idPlayer} **PRIMARY KEY**
- idMatch **FOREIGN KEY Match**
- idPlayer **FOREIGN Key Player**

Event
- idEvent **PRIMARY KEY**
- idMatch **FOREIGN KEY Match**
- idPlayer **FOREIGN KEY Player**
- minute **NOT NULL, CHECK (minute > 0)**
- idPart **FOREIGN KEY Part**

Part
- idPart **PRIMARY KEY**
- part **NOT NULL**

Goal
- idGoal **PRIMARY KEY; FOREIGN KEY Event**
- self **DEFAULT FALSE**

Card
- idCard **PRIMARY KEY; FOREIGN KEY Event**
- is_red **NOT NULL**

Penalty
- idPenalty **PRIMARY KEY; FOREIGN KEY Event**

Substitution

- idSubstitution **PRIMARY KEY; FOREIGN KEY Event**
- {idMatch, idPlayer} **FOREIGN KEY MatchPlayer**

BallPossession

- {idMatch, idClub} **PRIMARY KEY**
- idMatch **FOREIGN KEY Match**
- idClub **FOREIGN KEY Club**
- percentage **CHECK (0 <= percentage <= 100)**

Stadium

- idStadium **PRIMARY KEY**
- name **NOT NULL**

Queries

1. Shows a table with the ID, name, foundation year, group, stadium name and coach name of all the Portuguese teams

2. Shows a table with the ID, name, birthdate, and the total goals of all players who scored goals

3. Shows a table that have, for each country, the percentage of clubs that have that nationality

4. Shows a table that, for each stadium that, the quantity of players that had played in that stadium in this competition

5. Shows a table that has the number of substitutions that each on of the trainers have made

6. Shows the average quantity of players the clubs have

7. Shows a table with, for each team, the oldest and the youngest players (and their ages)

8. Shows the result of a specific match (In this case, we used the match Napoli-Ajax)

9. Creates a table that shows the money that each one of the clubs had one in the competition. Note: Matches that had not happened are considered a Draw

10. Creates a table with the results of a specific group (in this case, we used the group A)

Triggers

First Trigger - The Stadium

This trigger is used every time the stadium is not indicated when inserting a match. If the match isn't the final game, it will assume that the stadium of the match is the stadium of the Visited club. If the match is the final, the trigger will abort the insertion and will print the message: "Final - Need a Stadium".

This trigger makes total sense because it reflects exactly how the competition works. Usually, the stadium of the match is the stadium of the visited club, except for the final, where it can be any stadium.

Second Trigger - The Referee

Every time the referee of the match has the same nationality of one of the clubs playing, this trigger will print a message error: "Referee has the same nationality as one of the teams".

One more time, this trigger is useful because it comes from UEFA rules.

Third Trigger - The Group

If we try to add a match of group stage in the database, both teams need to belong to the same group, if that does not happen, we will receive an error message: "In Group Stage, both teams must belong to the same team" and the match won't be added to the database.

Participation of each member of the group

The elaboration of this delivery number two of the project elaborated in the context of the Databases curricular unit aimed to give continuity to the project already started. In this way, the objective to be fulfilled would pass through interrogating the database with different degrees of complexity already populated in delivery number one, as well as adding triggers to the database, useful for monitoring and maintaining the database.

The elements of this working group decided in a first phase to divide tasks: two elements worked on the elaboration of the queries for interrogating the database while another element developed the appropriate triggers for the context of the project. After a first phase of developing these tasks, the working group met to discuss what each one had done. In the development of the queries, the group shared a lot of ideas to make the developed queries increasingly complex and robust. Several meetings were held so that everyone was within the context of the problem.

In conclusion, each element of this working group considers that the commitment to the implementation and elaboration of this project that is now being delivered was equally carried out by all.