Relational and NoSQL Databases

Advanced Databases/Databases Technologies

2024/2025 Project – 1st Checkpoint

1. Group Details:

Group number: 8

Members: Afonso Baptista 58213; Miguel Borges 58187; Miguel Dinis 58198; Rafael Correia 58256

Every member of the group contributed equally to the development of this project.

- Afonso Baptista:
  - SQL – create and connect the database, create tables and insert data while keeping the relational schema in mind.
  - Relational schema – Entity-Relational model, Relational diagram and CREATE TABLE statements.
- Miguel Borges:
  - First simple query in Mongo and SQL:
    - *"Games where Portugal scored more than 3 goals after 2003."*
  - First complex query in Mongo and SQL:
    - *"Change the Neutral field to True for the matches that have more than 5 goals scored and that both teams have played with each other at least 100 times."*
- Miguel Dinis:
  - Second simple query in Mongo and SQL:
    - *"Games where Australia and Switzerland played against each other."*
  - Second complex query in Mongo and SQL:
    - *"Get the 100 matches with the most difference between the home score and away score and add 100 matches with the scores flipped, without shootouts and goal scorers."*
- Rafael Correia:
  - Data cleaning process.
  - MongoDB – Create and connect to the database, define the schema validator for the collection, create the collection itself, and insert the data while maintaining the structure of the collection.

Each member of the group wrote his contribution in the report.

2. Project Description

2.1. Phase 1: Data Modelling and Querying

Dataset Description

The dataset we chose is a Kaggle dataset (link to the dataset is available at the end of the report) on international football match results, goal scorers and shootouts. It includes 47917 results of international football matches from 1872 up to 2024. The matches are strictly men's full internationals, and the data doesn't include Olympic Games or matches where at least one of the teams was the B team, U-23 or a league select team. This dataset is divided into three files:

- *results.csv*: has a list of match results. Includes the following columns:
  - *date*: date of the match.
  - *home_team*: the name of the home team.
  - *away_team*: the name of the away team.
  - *home_score*: full-time home team score including extra time, not including penalty-shootouts.
  - *away_score*: full-time away team score including extra time, not including penalty-shootouts.
  - *tournament*: the name of the tournament.
  - *city*: the name of the city/town/administrative unit where the match was played.
  - *country*: the name of the country where the match was played.
  - *neutral*: whether the match was played at a neutral venue.

- *goalscorers.csv*: has the list of scorers in the games. Includes the following columns:
  - *date*: date of the match.
  - *home_team*: the name of the home team.
  - *away_team*: the name of the away team.
  - *team*: the name of the team that scored the goal.
  - *minute*: the minute the goal was scored.
  - *scorer*: name of the player that scored the goal.
  - *own_goal*: whether the goal was an own goal.
  - *penalty*: whether the goal was a penalty.

- *shootouts.csv*: has the list of winner teams of penalty-shootouts. Includes the following columns:
  - *date*: date of the match.
  - *home_team*: the name of the home team.
  - *away_team*: the name of the away team.
  - *winner*: winner of the penalty-shootout.
  - *first_shooter*: the team that went first in the shootout.

Observations

- In *goalscorers*, some entries lacked the *minute* field, leading to their removal.
- The *first_shooter* in the table *shootouts* had roughly 64% missing values, requiring imputation.
- *data* fields were stored as objects instead of datetime formats.

Data Cleaning Process

- Removing missing values: rows missing the critical *minute* value in *goalscorers* were dropped as they could not reliably represent goals.
- Imputation: missing *first_shooter* values in the *shootouts* table were filled with a random selection between the *home_team* and the *away_team*. These values are not critical, and their imputation ensures that the data is complete without biased results.
- Data type conversion: *date* values were converted into actual datetime objects for consistency and to enable time-based filtering.

2.1.1. Relational Schema

The data was modelled into three core entities:

- Results: Represents matches with the attributes of the dataset and another attribute named *match_id*, which is the Primary Key
- Goalscorers: Tracks individual goals scored during matches with the same attributes as the dataset, except for *away_team* and *home_team*, and adding two attributes named *id*, which is the Primary Key and *match_id,* which is the Foreign Key that points to *results.match_id*
- Shootouts: Records penalty shootout details for matches with the same attributes as the dataset, except for *away_team* and *home_team*, and adding two attributes named *id*, which is the Primary Key and *match_id,* which is the Foreign Key that points to *results.match_id*

These entities are linked through the *match_id* key, ensuring data consistency and enabling effecting querying across the database.

We decided to use three separate tables instead of one because that is what makes sense with the database design principles like normalization, scalability and maintainability.

By splitting the data into three core entities (*Results, Goalscorers, Shootouts*), the schema ensures that each table focuses on a specific aspect of the data used. Match-specific data is stored in the *Results*, while individual goal records and shootout details are stored in the *Goalscorers,* and *Shootouts,* respectively.

Separating the data into these three tables allows for easier updates and expansion of the database, and if the project expands to include additional features, such as player statistics of tournament summaries, these can be integrated into the schema without modifying the existing tables significantly.

The use of foreign keys between the tables also enforces referential integrity, ensuring that no goal or shootout can be associated with a non-existent match.

Not only that but using several tables gives us improved query performance since using only one table would result in a large, unwieldy table with many rows and columns. Queries that only require data from the *results* part of the data would have to scan through irrelevant data about goals and shootouts, slowing down performance.

2.1.2. MongoDB Collections

For the MongoDB collection structure, we decided to create a single collection containing the following: For each result from the CSV, we associate a list of goalscorers by date, home_team, and away_team. This list includes all columns that are not shared with the result. If a result does not have associated goals, the list remains empty. Additionally, we associate a shootouts object, which follows the same logic as goalscorers. If no shootouts are associated with the result, this field is set to null.

We chose to store all data in a single MongoDB collection because this non-relational database model is highly optimized for working with embedded data. By using this approach, we reduce the number of lookups and joins, improving query performance, especially in scenarios where related data is frequently accessed together. Moreover, by defining a schema validator, we ensure data integrity and maintain a consistent structure, even in a flexible environment like MongoDB. This decision allowed us to simplify the database design and optimize access to the most relevant information.

3. References

Original dataset's link: https://www.kaggle.com/datasets/martj42/international-football-results-from-1872-to-2017