Group 66

DataBase NBA

Fengfan Bian 7924160

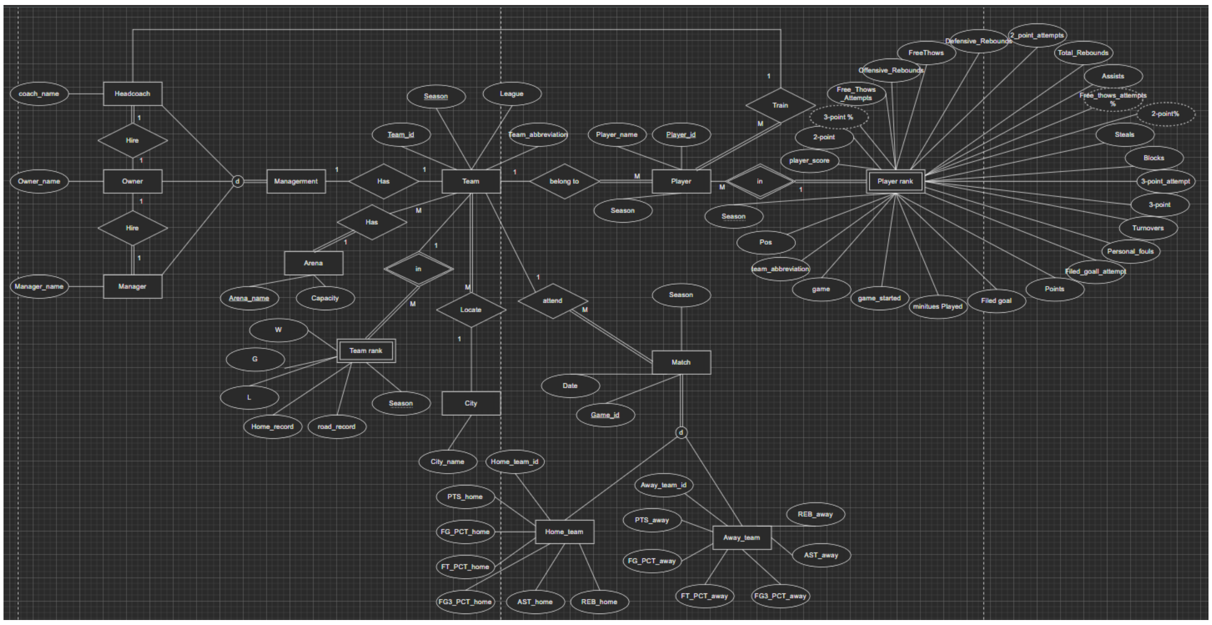
Wanqi Li 7908738

Xin Nie 7640563

Content

Part 1: Dataset

Summary of the data



Our first choice of the dataset was the COVID-19 data in Canada and the United States of America, however, we found that the metrics of COVID -19 in Canada and the USA are very different. For example, the USA has data updated every week, but Canada counted it every day. The USA has very specific metrics on how COVID affects different ages, which we were very interested in, but Canada doesn't have such data. In the end, we decided to change our dataset. All three of us are fans of basketball, therefore we chose the NBA as our dataset.

The data set consists of NBA teams' and players' information in seasons 2018 and 2019. For the teams' information, the dataset includes the teams' names, owners, coaches and kinds of relevant data. What's more, it also includes the team rank during these two years. For the players' information, the dataset includes the players' name and their teams for the respective year. In addition, it also includes the player rank during these two years.

The number of records we used is:

- 30 teams:
    - 30 owners
    - 30 coaches

    - 30 managers
- 29 cities for these 30 teams
- 29 arenas for these 30 teams
- 60 columns of team rank in seasons 2018 and 2019
- 1946 matches in seasons 2018 and 2019
- 1374 rows of data for players, some of them are the same person with different player IDs since we have to consider some player had changed their teams between 2018 and 2019. Then we need to use unique player IDs to identify the same player in a different year.
- 1039 lines of data for the players' rank. For this dataset, the NBA players mustn't be in the player rank but a player in this rank must be an NBA player.

Discussion of the data model

- We merged the schema first by the primary of each relation. The relations that have the same primary key were merged. 1 to many relationships, many-side can determine 1, many to many relationships need the primary key from both sides as a primary key.
- After the merge, we apply the 1st Normalization, 2nd Normalization, 3rd Normalization and BCNF to get the final schema.
- The difficult part was generating unique player IDs for players and using these player IDs on the other tables. E.g. the order in the player rank table is not the same as the player's table. But it can be completed by using R studio.
- Most CSV files from the website still need some changes. Some of the files use different styles to represent seasons. E.g. 202019 and 2019. However, this problem can be fixed by R.

Part 2: Query

After we normalized our functional dependencies we conclude ten tables and put them in words to create tables. The challenging thing is we have to input suitable variable types that the SQL server can run, which is also the part I learned the most. For example, learned how to use "decimal()" to define variables, the "numeric" type is limited not to be a primary key and the "text" type of variables cannot "order by" so we used “varchar()” with the size limit to instead to keep the data for accurate and safe. Some of the primary keys of the weak entity need to be set up "on delete cascade". After that, we used a python program that made ourselves automatically convert our format CSV files into "insert into" statements.

List of Queries

In Search Player:

1, Search player by a given player name.
2, List of players in one team in a selected season
3, Match a player played in a selected season, search by entering a player name, searching result includes data of the match, the home team of this match and the away team of this match.

In Search Team:

1, List all teams in the league
2, Search team by giving a team name
3, List all Team's WinRate in a given season.

In Search Performance:

1, Top 5 players in a selected team by selected field(such as 3-point shoots, how many blocks the play gets, etc)

2, Top 15 Team by selected field(such as 3 point shoots, how many blocks the play get,etc)

3, Top 15 Team by hit rate(2-point or 3 point).

4, The advantage of different positions on the court in the selected field and a selected season (such as which position on the court is good at shooting 3 points).

The interesting queries we made contain almost all the features.
For searching Players, we have searched the player by given string and list of all players in one selected season. For finding the Matches details of a player search by a String we used "Union" to connect the matches that happen in the home team and away team then get all the matches of that player.

For the search of teams, we have searched the team by given string and list of all teams and their information in one selected season. For getting the win rate of each team for a different season, we used the number of win matches to divide its total matches and order them in descending order.

In the performance part, we can get the top 5 players in a team by the selected field which uses "order by" to order the given field in descending order and limit it to the top 5. To find the Top 15 Team by selected field(such as 3 point shoots, how many blocks the player getting, etc) use "with" to create the connection between team property(team_id) with player rank(the total score), "SUM()" function and group by functions to define the total score of each team, then we order them in descending order and pick the top 15. To find the Top 15 Team by hit rate(2 point percent or 3 points percent) in different seasons we take the "AVG()" of the rate of a given field and connect it with team property to get each team score separately, finally get relevant team information and its rank. For having the advantage of different positions on the court in the selected field and selected season(such as which position on the court is good at shooting 3 points) used the "AVG()" function and "GROUP BY" to get which position did the highest score on that field.

Part3 GUI

The Graphical User Interface part is developed in Java. Swing is a GUI widget toolkit for Java, which is used to create window-based applications. In this project, some main components are from javax.swing package that we used are:

- JFrame was used to create a window.
- JMenuBar, JMenu, and JMenuItem were used to create the menu bar and its corresponding selection items.
- JPanel was used to create separate panels in each frame in the window.
- JTextField, JCheckBox, and JComboBox are used to collect user input and pass input to the database.
- JButton was used to generate a button in the window to submit a search requirement to the system.
- JTable was used to display the search result to the user in a table format.

In general, the GUI was developed based on what kind of queries we had for the database. For each query, there is a corresponding window/panel in the GUI for the user to enter their input, and click on the button to start searching for results in the database. For example, in the Search Player, we have a query to display a list of players from a selected team by a selected season. A new window will be popped by clicking the submit button, which contains a list of players from the selected team in the selected season.

Some problems that we met when developing the GUI. First, in the beginning, we used the JTextArea as the main display port, and we spent lots of time formatting the output. However, the output was not desired. Until we found that javax.swing package has a JTable component which can directly output the java ResultSet as a table to the window, which is the way we finally used it in this project. Secondly, we had a big issue when adding a menu to each window, whenever the application opens a new window to display the search result, the menu in the mainPage window whether not respond, or its menu items are displayed multiple times. After a very long time debug, we found that whenever a new window is created, the program repeatedly calls the create menu method, which causes duplicate menu items to be created. Therefore, we created a new Menu class and every window could add a menu object to itself, which solved the problem. Thirdly, we added the functionality of exporting the table as a CSV file, then we could generate some graphs based on the file. However, we didn’t have enough time to finish it, which is a pity.

Part 4 Reference

Basketball Reference: **https://www.basketball-reference.com/leagues/**

NBA games data: **https://www.kaggle.com/datasets/nathanlauga/nba-games**