Database Project

Prepared for:
Dr. Jalal Omer

Prepared by:
Elyas Belkhir
Samuel Benicewicz
Luca Donadello
Dylan Farmer
Mark Olson
CS 4347.003

December 3, 2023

TABLE OF CONTENTS

INTRODUCTION

Our project aims to provide users with crucial information regarding NBA players and teams. In particular, we developed the database system and web application to enable users to retrieve and modify data faster and more efficiently. Below is a detailed description of the system we have developed.

The NBA database system is an application that allows users to search and collect data for statistical data-gathering purposes easily. Users will also be able to narrow their search by filtering results. Our database system promotes program-data independence between our search query interface and the central database. Using a database approach, we Supported relations between various aspects, such as players, matches, and teams, to manipulate and store data. A particular Issue solved with our application is concurrent access in querying and updating the database. In particular, our system allows for simultaneous updates, including security measures to enforce data correctness. In summary, we implemented relationships between our data, solved issues using a database, and provided security to the data using the database system approach.

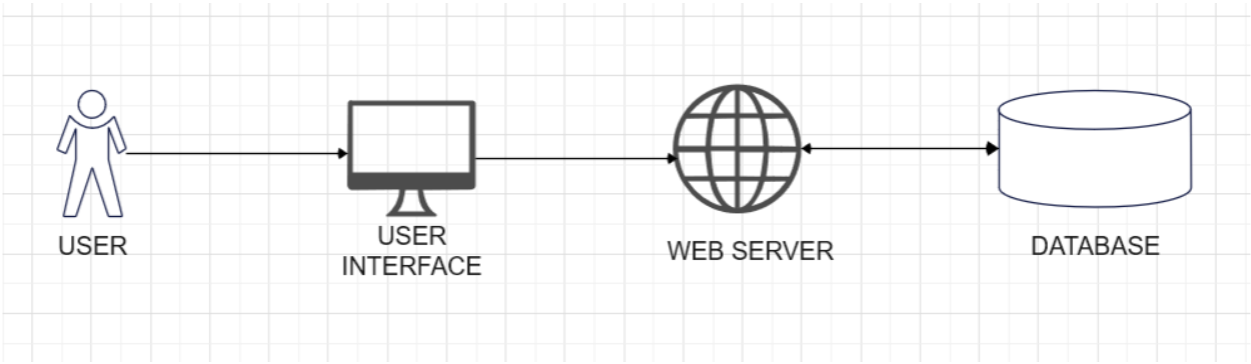
The target users for this application include basketball team managers, sports analysts, and sports fans. Basketball team managers, in particular, would greatly benefit from utilizing our system. They may be looking to have someone transfer to their team, want to monitor their player performances or see how they rank up compared to the other teams. Sports analysts and fans would also immensely enjoy accessing our database of basketball players. It will allow sports analysts to provide accurate information to the people they talk to and predict how the players will perform in future games. The more accurate their information is, the better they do. Sports fans will also be interested in keeping up with their favorite team and players. Many sports fans like to keep track of the statistics of their favorite teams and players and talk about the statistics with others. The platform we created has real-world applications for many users. It makes it simple for basketball team managers, sports analysts, and sports fans to quickly and easily view the statistics of any basketball player they search.

In our application, we implement an easy and efficient web interface. In particular, it enables the users to retrieve, update, delete, and manage data inside the database. The following overview summarizes the essential elements we implemented in our web interface.

The interface is user-friendly and easy to use. The user is prompted with different options to manage or retrieve specific information they are searching for inside the database. This follows the standards of a “webform” with labels and input boxes that permit the user to perform actions such as retrieving, manipulating, or adding data inside the database. Our implementation can also detect and handle errors derived from input errors from the user, simply providing a message to the user with the relevant error. The web interface maintains the data relative to the table “hidden” to the users and communicates with the database to change any information. It will display only the information the user intends to receive in a friendly and easy-to-view format. It also maintains a high level of security, preventing hacking attempts such as injection using the prevention method named prepared, where the input is first stored and analyzed and then sent to the database.

SYSTEM REQUIREMENTS

Below is a simple implementation of the context diagram, which provides a simple graphical view of our developed application. As can be seen, the user interacts with a web interface that retrieves data from a database. The database also communicates with the web interface to modify the data stored inside the database. In particular, the information will be available to the user thanks to the interaction between the request from the web interface and the database system. Then, the database system will produce, according to the request, the data to display to the user if necessary.



Interface Requirements

The application has the following interface requirements for the system.

User-Friendly Dashboard:

A User-friendly dashboard offers a simple and easy-to-use interface for users to get essential data and carry out routine operations.

A well-designed dashboard facilitates easy feature access, efficient application navigation, and viewing of pertinent data. It should provide key performance indicators in an easy-to-read style.

Responsive Web Interface:

Ensure the web interface is accessible and responsive across various devices, such as tablets, smartphones, laptops, and desktop computers.

The application may be used by NBA pros and enthusiasts from various devices and places, enhancing accessibility and usability; the responsive design guarantees a consistent and ideal user experience across several screen sizes.

Search and Filter Functionality:

Provide robust search and filter features so users can quickly find specific information in the database, such as looking for a specific player, team, or game.

An NBA database contains a large quantity of data, and users may quickly and easily narrow down results and locate pertinent information by using effective search and filter capabilities. This holds special significance for users who must do focused searches or analyses.

Real-Time Updates:

Real-time information is essential in a dynamic setting such as professional sports. By ensuring users are up to date on the most recent scores, events, and modifications, data updates improve both the overall user experience and the use of the program.

Functional Requirements

Below is a list of the functional requirements our system aims to maintain.

Player Data Administration:

The database should allow the storage and retrieval of comprehensive player data about NBA players quickly and easily. Personal information such as name, date of birth, performance data points per game, and other pertinent player-related information are all included in this.

Tracking Game Statistics:

Every NBA game should be able to be tracked and recorded by the system in great detail. This contains data on team statistics, player performances, game length, scores, and other pertinent variables. The database should support real-time updates throughout the game and the archiving of previous data for reporting and analysis.

Performance & Scalability:

Over time, as data volume increases, the database must be scalable to accommodate higher demands without sacrificing efficiency. This entails indexing, query optimization, and considering horizontal scaling-supporting technology. The system must be able to manage several users' simultaneous access with efficiency and offer a responsive user interface.

Data Consistency and Integrity:

To guarantee the quality and dependability of the information, the database should enforce data consistency and integrity. This calls for the use of constraints, validations, and transactions to avoid mistakes, inconsistencies, and duplicate entries. Maintaining data quality is aided by using normalization techniques and a well-designed database schema.

Non-Functional requirements:

Below is a list of the non-functional requirements our system aims to maintain.

Usability:

The responsiveness of the user interface should guarantee that tasks such as requesting player data and switching between pages are accomplished in an average of three seconds.

Scalability:

Build the database and application architecture to accommodate future expansion in more NBA clubs, players, and seasons. The system should handle a fifty percent increase in data volume with grace and without causing any appreciable performance degradation.

Extensibility:

Modularity and extensibility should be considered throughout the program design process to facilitate the simple integration of additional functionality or data types. This includes adding more multimedia material, broadening team information, and integrating new data categories without requiring significant code changes.

Resource Usage:

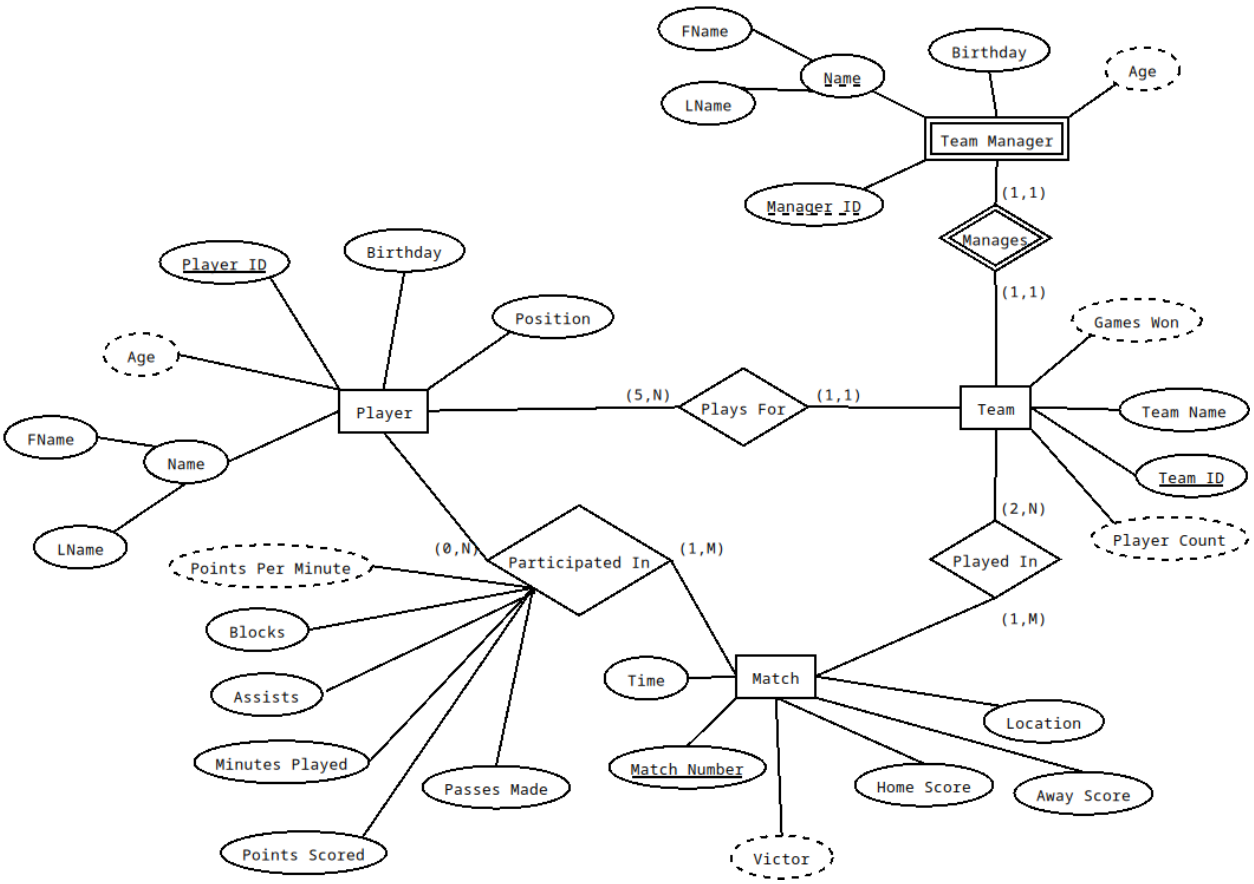
The program must be optimized to make the most use of system resources. To guarantee responsiveness and smooth operation, keep the server's CPU and memory utilization below 80% during periods of high demand.

Data Retrieval Speed:

To provide a smooth user experience, database queries for frequently requested statistics reports, such as player averages, should be planned and optimized to produce results in less than three seconds.

CONCEPTUAL DESIGN OF THE DATABASE

ER Diagram



Originally, Manager ID and Team ID were not attributes, but were added in later for the functionality of the database implementation's update and delete.

Some rules:

- A team includes at least 5 players. In the database implementation itself, all teams consist of five players.
- A player can only play for one team at a time. Unlike the NBA database we originally took inspiration from, the player's team can be retroactively updated (instead of a new tuple of the same player name added to account for the team change).
- Players don't have to participate in a match if their team plays in it. Implementation-wise, that would mean that their associated Participated In tuple will be zeroed out.
- As can be derived from the ER diagram, a match can be held without any players participating in it. As unrealistic as it sounds, the design makes it possible.

- Each match must have two participating teams, home and away. Any one team can play in an unlimited amount of matches.
- A manager can only manage one team at a time.

Database Dictionary

Player		
Column	Data Type	Description
Player ID	int	Player’s unique numerical ID
FName	varchar(255)	Player’s first name
Lname	varchar(255)	Player’s last name
Birthday	date	Player’s birthday
Position	varchar(20)	Player’s position on team

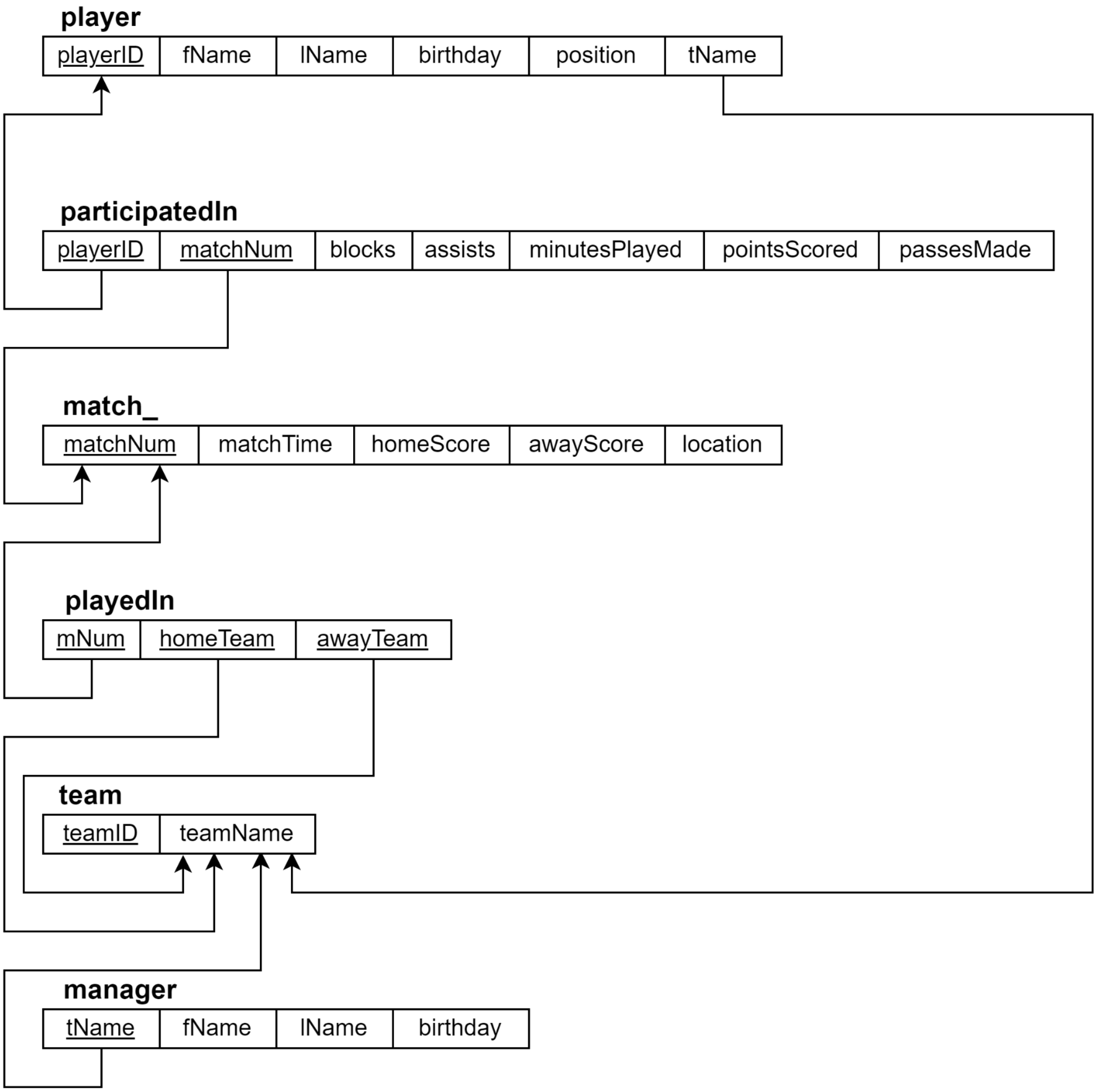
Team		
Column	Data Type	Description
Team ID	int	Team’s unique numerical ID
Team Name	varchar(255)	The name of the team

Manager		
Column	Data Type	Description
Manager ID	int	Manager’s unique numerical ID
FName	varchar(255)	Manager’s first name
LName	varchar(255)	Manager’s last name
Birthday	date	Manager’s birthday

Match		
Column	Data Type	Description
Match Number	int	Match’s unique numerical ID
Time	time	Duration of the match
Home Score	int	Home Team’s final score
Away Score	int	Away Team’s final score
Location	varchar(255)	Location the match took place

Participated In		
Column	Data Type	Description
Blocks	int	Number of blocks a player made in a match
Assists	int	Number of assists a player made in a match
Minutes Played	int	Minutes played in a match
Points Scored	int	Points scored during a match
Passes Made	int	Passes made during a match

LOGICAL DATABASE SCHEMA



player

playerID	fName	lName	birthday	position	tName

participatedIn

playerID	matchNum	blocks	assists	minutesPlayed	pointsScored	passesMade

match_

matchNum	matchTime	homeScore	awayScore	location

playedIn

mNum	homeTeam	awayTeam

team

teamID	teamName

manager

tName	fName	lName	birthday

The Following SQL Statements are used to construct the database schema:

```
CREATE TABLE team(
        teamID INT NOT NULL AUTO_INCREMENT,
        teamName VARCHAR(255) UNIQUE,
        CONSTRAINT PK_team PRIMARY KEY (teamID)
);

CREATE TABLE manager(
        managerID int not null auto_increment,
         tName varchar(255) not null,
        fName varchar(255),
        lName varchar(255),
        birthday date,
        constraint PK_manager primary key (managerID),
        constraint FK_managerTname foreign key (tName) references team(teamName)
        on delete cascade on update cascade
);

CREATE TABLE player(
        playerID int not null auto_increment,
        fName varchar(255),
        lName varchar(255),
        birthday date,
        position varchar(20),
        tName varchar(255),
        constraint PK_player primary key (playerID),
        constraint FK_player foreign key (tName) references team(teamName)
        on delete cascade on update cascade
);

CREATE TABLE match_(
        matchNum int not null auto_increment,
        matchTime time,
        homeScore int,
        awayScore int,
        location varchar(255),
        constraint PK_match primary key (matchNum)
);
```

```
CREATE TABLE participatedIn(
        playerID int not null,
        matchNum int not null,
        blocks int,
        assists int,
        minutesPlayed int,
        pointsScored int,
        passesMade int,
        constraint PK_participatedIn primary key (playerID,matchNum),
        constraint FK_participatedIn_id foreign key (playerID) references player(playerID)
        on delete cascade on update cascade,
        Constraint FK_participatedIn_matchNum foreign key (matchNum) references match_(matchNum)
        on delete cascade on update cascade
);

CREATE TABLE playedIn(
        Mnum int not null,
        homeTeam varchar(255) not null,
        awayTeam varchar(255) not null,
        constraint PK_playedIn primary key (Mnum,homeTeam,awayTeam),
        constraint FK_playedIn_matchNum foreign key (Mnum) references match_(matchNum)
        on delete cascade on update cascade,
        constraint FK_playedIn_home foreign key (homeTeam) references team(teamName)
        on delete cascade on update cascade,
        constraint FK_playedIn_away foreign key (awayTeam) references team(teamName)
        on delete cascade on update cascade
);
```

The expected database operations could include creating, retrieving, updating, and deleting the contents of all tables, to match the current NBA information to be represented within the database. All of which can be achieved through the accompanying database webpage.

The estimated volume of data being stored within the database at any given time will vary based on how long the database has been operated and how much information has been added/removed over time. And could also vary based on the current regulations and activities characteristics of the NBA league as a whole. But a loose estimate can still be derived for each table respectively based on the expected number of entries within each table for a given sport season. For example, if there are expected to be 1000 matches in a given season and there are currently 30 existing teams with a maximum of 21 players at any given time. We can estimate that the participatedIn table will contain a maximum of 630,000 entries(1000 x 30 x 21 = 630,000). Based on these parameters the match_ and playedIn tables will contain a maximum of 1000 entries each(if there are 1000 games), the player table will contain a maximum of 630(30 x 21 = 630) players(no more than 21 players per team), and the team and manager tables will both contain 30 entries(one per team).

FUNCTIONAL DEPENDENCIES AND DATABASE NORMALIZATION

Tables

- Player(PlayerID, Fname, Lname, Birthday, Position, Tname)
- ParticipatedIn(PlayerID, MatchNum, Blocks, Assists, MinutesPlayed, PointsScored, PassesMade)
- Match(MatchNum, Time, Homescore, Awayscore, Location)
- PlayedIn(Mnum, HomeTeam, AwayTeam)
- Team(TeamID, TeamName)
- Manager(ManagerID, Tname, Fname, Lname, Birthday)

This relational schema is already in 1NF, but we need to introduce functional dependencies so that each non-prime attribute is fully dependent on a candidate key to satisfy 2NF. Thus, the following FD's will be introduced:

- Player
  - PlayerID → Fname, Lname, Birthday, Position, Tname
- ParticipatedIn
  - PlayerID, MatchNo → Blocks, Assists, MinutesPlayed, PointsScored, PassesMade
- Match
  - MatchNum → Time, Homescore, Awayscore, Location
- PlayedIn
  - Mnum → HomeTeam, AwayTeam
- Manager
  - ManagerID → Tname, Fname, Lname, Birthday
- Team
  - TeamID → TeanName

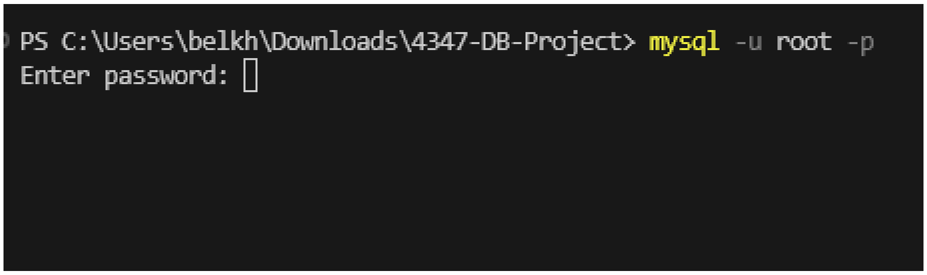
This schema also satisfies 3NF, as there is only 1 FD for each table that has one, making transitive dependencies impossible.

This schema also satisfies BCNF, as each FD's left hand side is a key of the table it corresponds to. This includes Mnum → HomeTeam, AwayTeam, as Mnum is inherently a key in and of itself due to every other attribute depending on it.

No fundamental changes to the database's structure were made, so the SQL statements for constructing it remain the same.

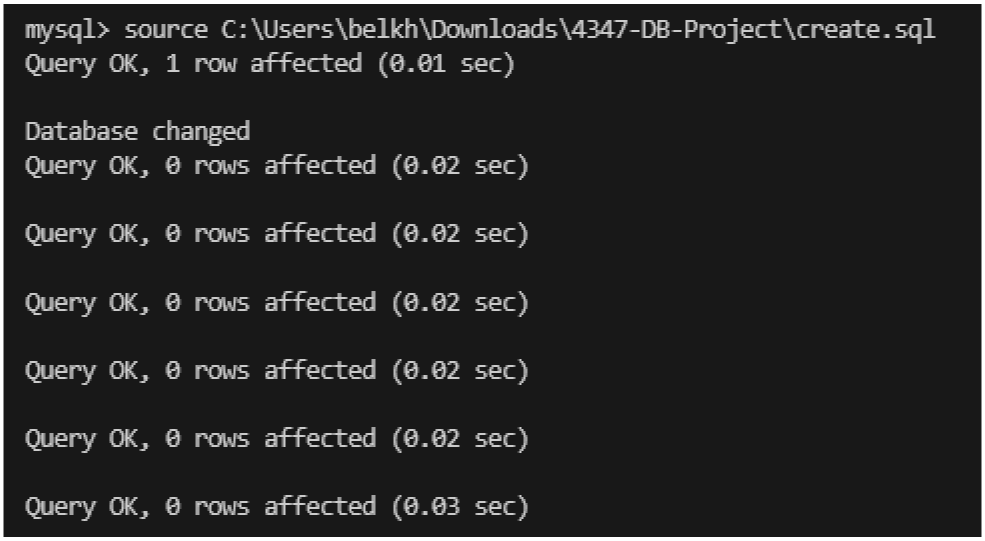
THE DATABASE SYSTEM

To install and invoke the database system, first install MySQL server from the MySQL website (https://dev.mysql.com/downloads/mysql/). Next, start the database server in the command line.



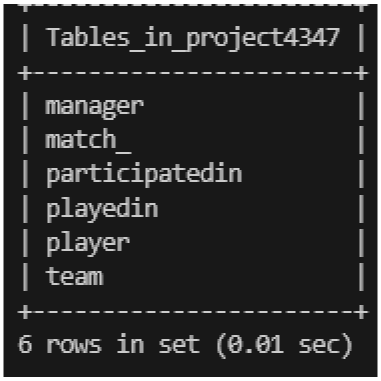
```
PS C:\Users\belkh\Downloads\4347-DB-Project> mysql -u root -p
Enter password: 
```

Next use the following command to run the create.sql file which instantiates the basketball database, switches to the newly created table, and creates all tables and constraints.



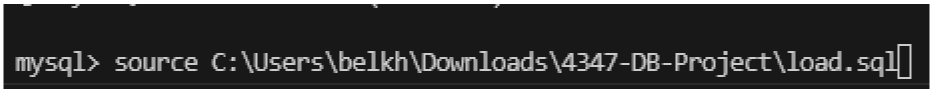
```
mysql> source C:\Users\belkh\Downloads\4347-DB-Project\create.sql
Query OK, 1 row affected (0.01 sec)

Database changed
Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.03 sec)
```

Now all tables are created and ready to be populated.



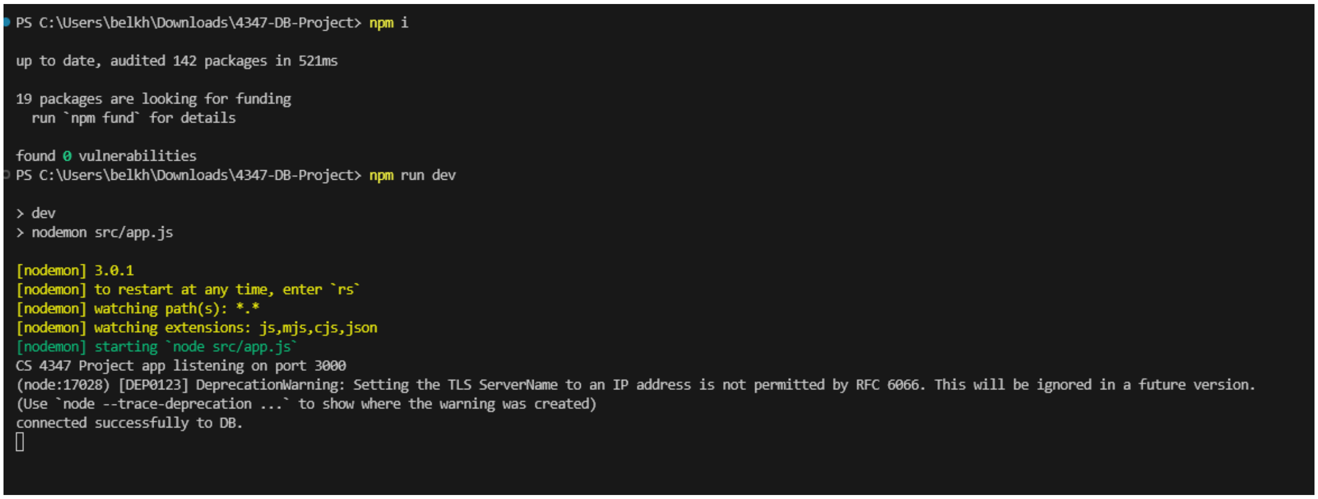
```
| Tables_in_project4347 |
+-----------------------+
| manager               |
| match_                |
| participatedin        |
| playedin              |
| player                |
| team                  |
+-----------------------+
6 rows in set (0.01 sec)
```

Next, run the load file to populate all data into each newly created table.



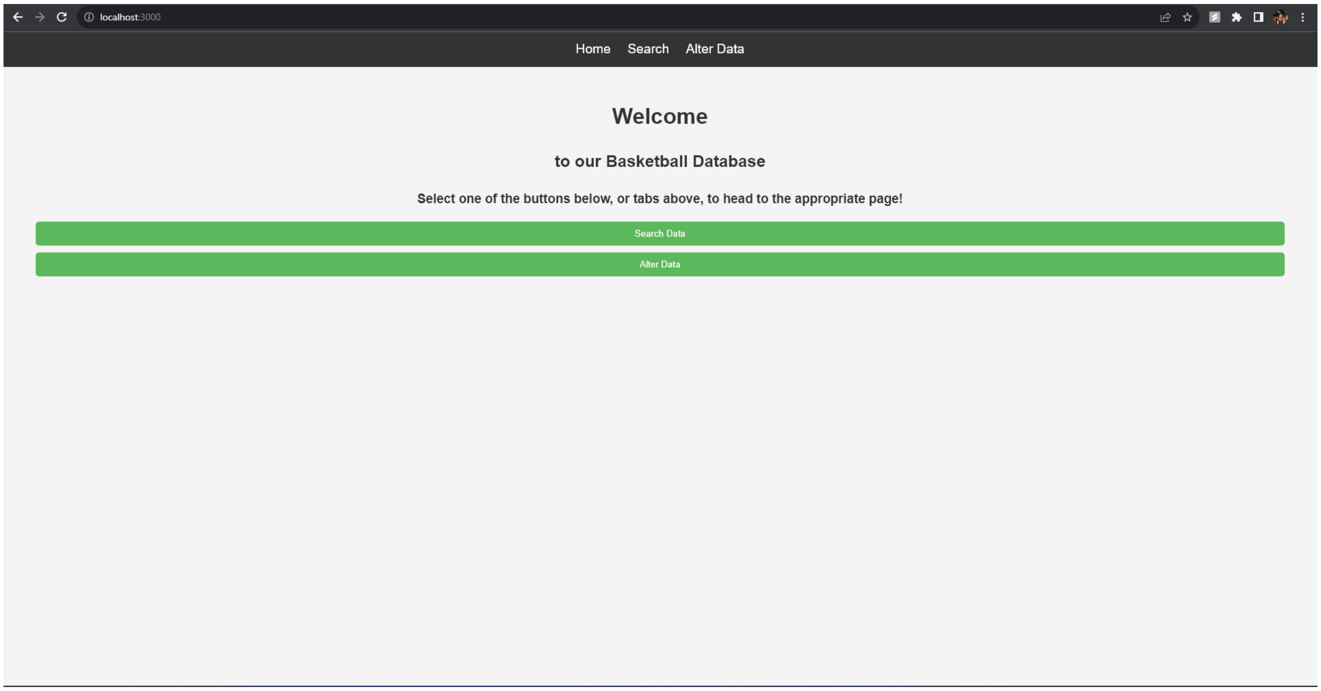
```
mysql> source C:\Users\belkh\Downloads\4347-DB-Project\load.sql
```

Now the data is loaded, and we can start the node.js web server. Download node.js (https://nodejs.org/en/download), then in the command line, run `npm install` followed by `npm run dev` as shown below to start the server that will communicate with the database following requests in the HTML frontend. Be sure to edit the db.js file in the project directory to change the user and password to your respective database server credentials.



```
PS C:\Users\belkh\Downloads\4347-DB-Project> npm i

up to date, audited 142 packages in 521ms

19 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\belkh\Downloads\4347-DB-Project> npm run dev

> dev
> nodemon src/app.js

[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node src/app.js`
CS 4347 Project app listening on port 3000
(node:17028) [DEP0123] DeprecationWarning: Setting the TLS ServerName to an IP address is not permitted by RFC 6066. This will be ignored in a future version.
(Use `node --trace-deprecation ...` to show where the warning was created)
connected successfully to DB.
```

Now head to whichever localhost port the server and application are running on to use the application with its full functionality.



Welcome

to our Basketball Database

Select one of the buttons below, or tabs above, to head to the appropriate page!

USER APPLICATION INTERFACE

The user application interface is a vital part of the system. It allows the user to have simple and meaningful interaction with the database. It needs to provide clarity for the user, as well as provide useful information, such as feedback if something is incorrect or if an action was successful.

Building the System

We built the user interface using HTML, JavaScript, and CSS. HTML was used to create the various buttons and text boxes that are seen throughout the website. JavaScript was used to give functionality to all the buttons and text boxes, as well as for collecting the information that users entered. CSS was then used to make everything look more comfortable and easier to see for the users. All of these elements together provided for a more defined user interface.

How to Use Our System

Our system was designed with simplicity in-mind for the users. We wanted to ensure that anybody could use our system without needing to look at additional instructions. When users first go to our website, they are greeted with a homepage that welcomes them and gives them simple options to choose from.

If a user selects to search data, they get moved to another page with a dropdown box. From this dropdown, they are able to select what type of search they would like to perform, such as for a player, a manager, a certain match, etc. Once they make a selection, text boxes with descriptions for that specific selection appear. This allows users to enter in their search criteria. We also allow for fuzzy-search, where users can enter any information they want, whether it's only a couple fields, or even partial text in a single field, and it will show all results that match their given criteria. The results are then displayed below in an easy-to-read table so the users can see the results and find what they are looking for.

If the user selected they wanted to alter the data instead, they are moved to another page that contains two dropdown boxes. The first dropdown box allows them to select what type of action they would like to perform (create, update, delete). The second dropdown prompts the user to select which table they would like to alter the data in. Once both selections are made, the text fields with descriptions would show up. The text fields are specific to the table selection that the user made. The data alteration works by entering in the ID number of the record you want to alter, then any information you input into the fields will be what changes in the record correlating to that ID number. If there is no match to an ID number that was entered, then the user will be displayed with an error message saying that it was an incorrect ID and to try again. If the action was successful, it will notify the user of this.

While on the Search or Alter screens of the webpage, users may also use ENTER to submit their currently entered form values or DELETE to clear all form fields quickly instead of clicking on the respective buttons. These keyboard inputs were included to increase functionality of the webpage and allow faster workflows for all users.

CONCLUSION AND FUTURE WORK

This system was designed with all aspects in mind of connecting the user to the database system and providing them with a clear and understandable way to access and alter information. This system is useful for people wanting to keep up with their current team stats, for managers possibly looking to recruit players from other teams, and even news anchors that want to provide relevant information in their talks. The database provides a simple-to-use interface for anyone of any skill to be able to use and access. Future work that can be done to provide even further improvements can include having team logos show up with results, allowing the ID numbers to be auto-generated rather than user-defined, auto-gathering new data through an API rather than having to manually enter it, and providing the option to search multiple tables at the same time rather than just one. These are improvements which we plan to continue with and make general improvements to, as this is a system we believe can be helpful to others.

REFERENCES

[1] R. Elmasri and S. Navathe, *Fundamentals of Database Systems*, 7th ed. Boston: Pearson, 2015.

[2] “MySQL documentation,” MySQL, https://dev.mysql.com/doc/. Accessed Dec. 3, 2023.

APPENDIX

- 4347-DB-Project.zip