Project Step 6 Portfolio Assignment

Basketball League Database

Team 175 – Space Balls

Members: Brandon Kim, Ahmed Mustafa

Project URL: https://nba-manager.vercel.app/

Table of Contents

Executive Summary

This database was initially designed for a large basketball league to manage players, coaches, and rosters. Over the course of the project, we stayed true to our original visions but adopted changes to our structure and which data elements were in our final database. These changes allows us to produce a final product was responsive, intuitive, and allows all CRUD functions. The project went through the process of database design, creation of the SQL database, frontend design, then finally we did an front end and backend overhaul when we shifted from MySQL/Node.js to SQLite/Next.js.

Actions based on Project Feedback

Initially, we sent out with the vision of having coaches, players, teams, rosters, and playoff rounds. While this goal was ambitious and loved by our peers, we found that it was significantly harder to implement. We saw much of this during the project feedback stages. Below, we have made a brief list of all major actions taken based on the feedback from the initial project to now.

- Our peers suggested reducing the players from 15 per team to 5. This would greatly help reduce the size of the database, while maintaining enough players for functionality.
- The removal of the *coaches* entity was suggest by our TA. While the coaches would have neem great, it exceeded the scope of the project. Therefore, we removed it to focus on the core of the project.
- Removing a player's position. We found this was an unnecessary element based on peer feedback. It would be best to implement if we had time later in the project.
- For normalization, we expanded on the removal of elements that were redundant. This predominantly applied to the *Playoff Round* column in the *Rosters* table. Since a team had to pass through the semifinals to get to the finals, we made this column only reflective of the team's last game of the season. This also moved removed the M:N for these items.
- Updates to *CASCADE* allowing deletion of playoff rounds. This would turn the last *Playoff Round* element in *Rosters* to NULL if the playoff round is deleted from the *Playoff Round* table.
- Search and filter functionality added to all tables.
- Reduced teams from 30 to 15. This simplified the database. 30 teams would give us 150 players per year. The removal allows us to focus more of the project scope.

Updates from Previous Drafts

- Moved from Node.js and MySQL to Next.js (REACT) and SQLite. Next.js gave us improved speed and handled full stack development better than Node.js.
- New player additions populate when creating or modifying roster(s).
- Improved speed and fluid operations by changing websites backend. Early versions displayed tabled with significant lag. The team removed redundant SQL requests and simplified table rendering inside of Typescript code to improve user experience.

Project Outline / Overview

A basketball league needs data infrastructure to manage teams and personnel dynamically throughout the year. Players, coaches, and teams need to record which personnel are currently with which teams. The league has 5 starting players per roster with 15 total teams. This can give us up to 75 players per season in the database. Additionally, the league needs prior data to see the history of the teams each year. For this database, we will include the current year, 2023/2024, and the past year 2022/2023. The 2023/2024 season will be referred to as 2023. This naming protocol will be followed for the proceeding years. This will allow us to see the tenure of each individual player in the league. Our database will create entities of each player, team, and roster. This will allow us to easily find, update, or delete items from this data as needed by the organization. Each roster's post-season performance will also be available. This data will contain either none or the furthest round that roster advances to. The database will be accessible from a frontend website that allows an organization to search for a team or player. Historical queries can also be initiated from this site to view a specific team's roster from within the last two years with the possibility to expand to additional years if the league desires.

Database Outline

- **Players**: Stores each NBA player's basic information
    - **playerID**: (int, auto_increment, unique, not NULL, PK)
    - **firstName**: (varchar(255), not NULL)
    - **lastName**: (varchar(255), not NULL)
    - Relationships:
        - M:N relationship between **Players** and **Rosters** through intersection table **RosterPlayers** using **playerID** as FK.
- **Teams**: Stores each NBA team's basic information
    - **teamID**: (int, auto_increment, unique, not NULL, PK)
    - **name**: (varchar(255), not NULL)
    - **city**: (varchar(255), not NULL)
    - **state**: (varchar(255), not NULL)
    - Relationships:
        - 1:M relationship between Teams and Rosters using **teamID** as FK.
- **Rosters**: Stores a team's roster for a specific season
    - **rosterID**: (int, auto_increment, unique, not NULL, PK)
    - **year**: (varchar(4), not NULL)
    - **teamID**: (int, not NULL, FK)
    - **playoffRoundID**: (int, FK) The last playoff round the roster made that season. Can be NULL.
    - Relationships:
        - M:N relationship between **Rosters** and **Players** through intersection table **RosterPlayers** using **rosterID** as FK.
        - M:1 relationship between **Rosters** and **Teams** using **teamID** as FK.
        - M:1 relationship between **Rosters** and **PlayoffRounds** using **playoffRoundID** as FK.
        - M:1 relationship between **Rosters** and **Coaches** using **coachID** as FK.
- **PlayoffRounds:** *Stores a rosters last postseason round played. Null will be for no postseason, and champion will be if they won it all.*
    - **playoffRoundID**: (int, auto_increment, unique, not NULL, PK)
    - **name**: (varchar(255), not NULL)
    - **Relationships**:
        - 1:M relationship between **PlayoffRounds** and **Rosters** using **playoffRoundID** as FK.

Entity Relationship (ER) Diagram

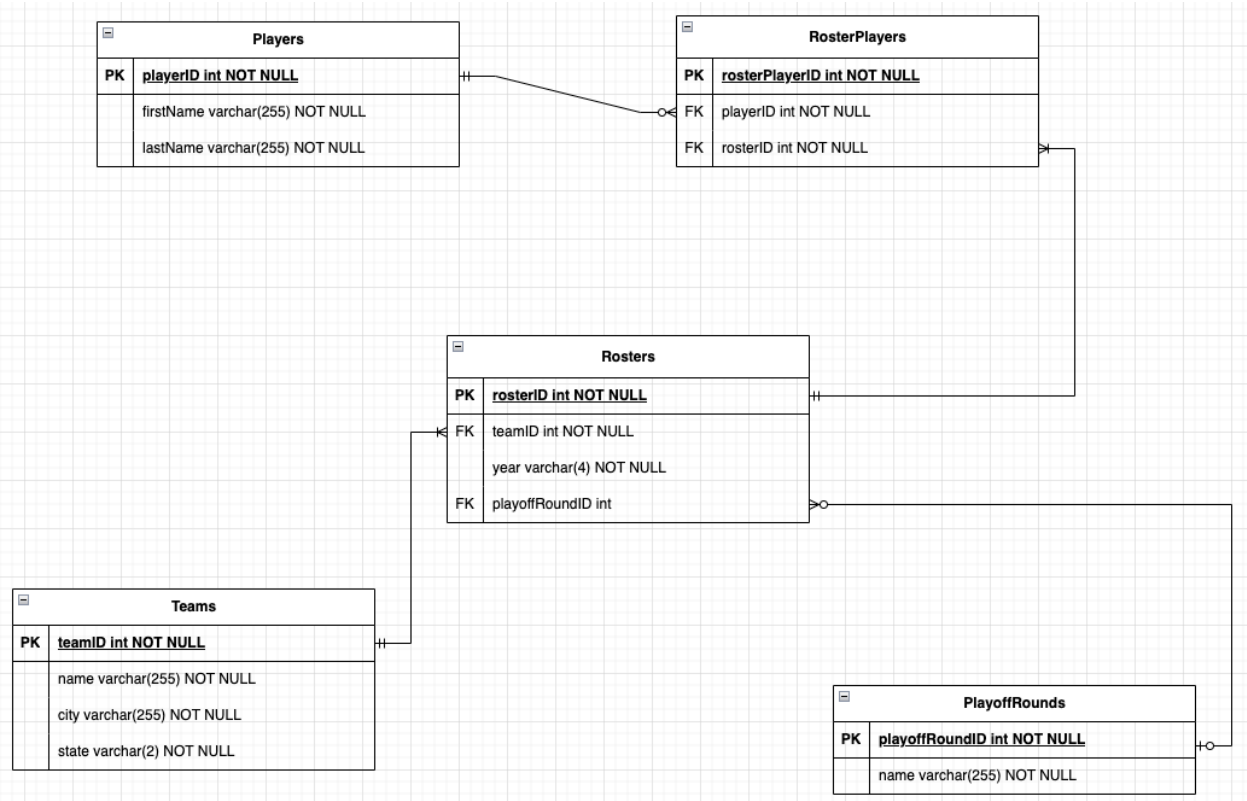


Figure 1: Entity Relationship Diagram for basketball league.

Schema

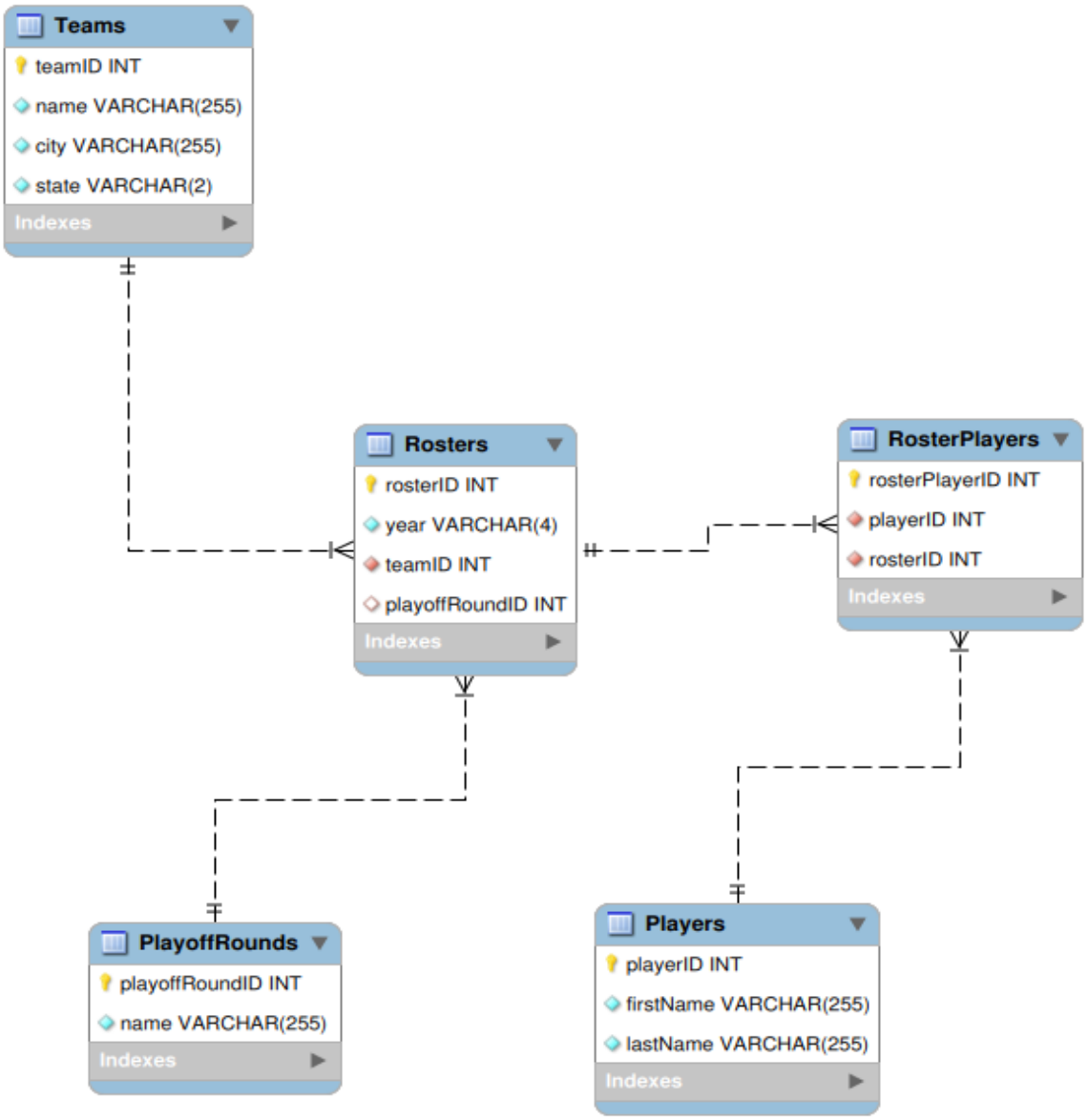


Figure 2: Schema for basketball database.

Sample Data Figures

	playerID	firstName	lastName
Edit Copy Delete	1	LeBron	James
Edit Copy Delete	2	D'Angelo	Russell
Edit Copy Delete	3	Austin	Reaves
Edit Copy Delete	4	Anthony	Davis
Edit Copy Delete	5	Rui	Hachimura
Edit Copy Delete	6	Jarred	Vanderbilt
Edit Copy Delete	7	Jrue	Holiday
Edit Copy Delete	8	Derrick	White
Edit Copy Delete	9	Jaylen	Brown
Edit Copy Delete	10	Jayson	Tatum
Edit Copy Delete	11	Kristaps	Porzingis
Edit Copy Delete	12	Marcus	Smart
Edit Copy Delete	13	Al	Hortford

Figure 3: Players Table

	playoffRoundID	name
Edit Copy Delete	3	Conference Finals
Edit Copy Delete	2	Conference Semifinals
Edit Copy Delete	1	First Round
Edit Copy Delete	4	NBA Finals

Figure 4: Playoff Round Table

	teamID	name	city	state
Edit Copy Delete	1	Clippers	Los Angeles	CA
Edit Copy Delete	2	Warriors	San Francisco	CA
Edit Copy Delete	3	76ers	Philadelphia	PA
Edit Copy Delete	4	Suns	Phoenix	AZ
Edit Copy Delete	5	Thunder	Oklahoma City	OK
Edit Copy Delete	6	Spurs	San Antonio	TX
Edit Copy Delete	7	Mavericks	Dallas	TX
Edit Copy Delete	8	Bulls	Chicago	IL
Edit Copy Delete	9	Nets	Brooklyn	NY
Edit Copy Delete	10	Wizards	Washington	DC
Edit Copy Delete	11	Knicks	New York	NY
Edit Copy Delete	12	Bucks	Milwaukee	WI
Edit Copy Delete	13	Hawks	Atlanta	GA
Edit Copy Delete	14	Lakers	Los Angeles	CA
Edit Copy Delete	15	Celtics	Boston	MA

Figure 5: Teams Table

	rosterID	year	teamID	playoffRoundID
Edit Copy Delete	1	2023	1	NULL
Edit Copy Delete	2	2023	2	NULL
Edit Copy Delete	3	2023	3	NULL
Edit Copy Delete	4	2023	4	NULL
Edit Copy Delete	5	2023	5	NULL
Edit Copy Delete	6	2023	6	1
Edit Copy Delete	7	2023	7	1
Edit Copy Delete	8	2023	8	2
Edit Copy Delete	9	2023	9	2
Edit Copy Delete	10	2023	10	2
Edit Copy Delete	11	2023	11	2
Edit Copy Delete	12	2023	12	3
Edit Copy Delete	13	2023	13	3
Edit Copy Delete	14	2023	14	4
Edit Copy Delete	15	2023	15	4

Figure 6: Roster Table

		rosterPlayerID	playerID	rosterID
☐	Edit Copy Delete	1	1	1
☐	Edit Copy Delete	2	1	2
☐	Edit Copy Delete	3	1	3
☐	Edit Copy Delete	4	1	4
☐	Edit Copy Delete	5	1	5
☐	Edit Copy Delete	6	2	6
☐	Edit Copy Delete	7	2	7
☐	Edit Copy Delete	8	2	8

Figure 7: RosterPlayer join table

UI Screen Shots with Informative Titles

Figure 8: CRUD for Rosters. CREATE/READ/UPDATE/DELETE Rosters.

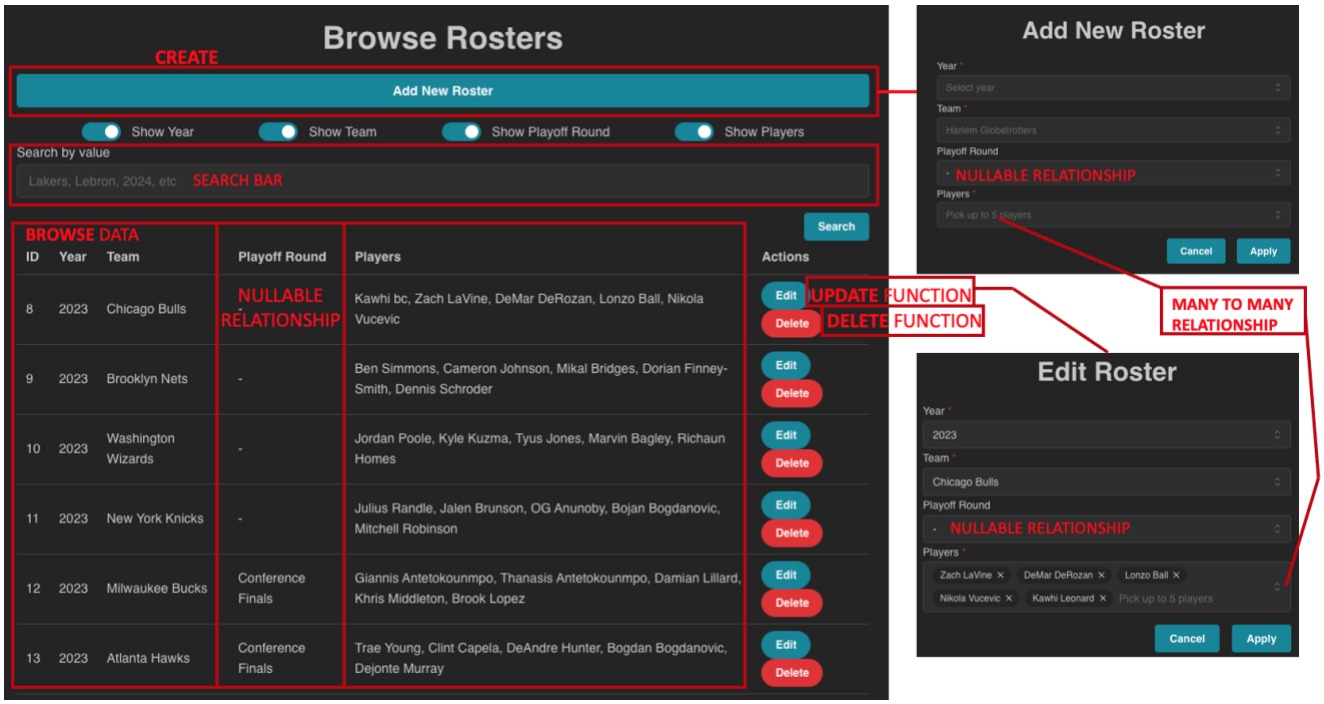


Figure 9: CRUD for Players. CREATE/READ/UPDATE/DELETE Players.

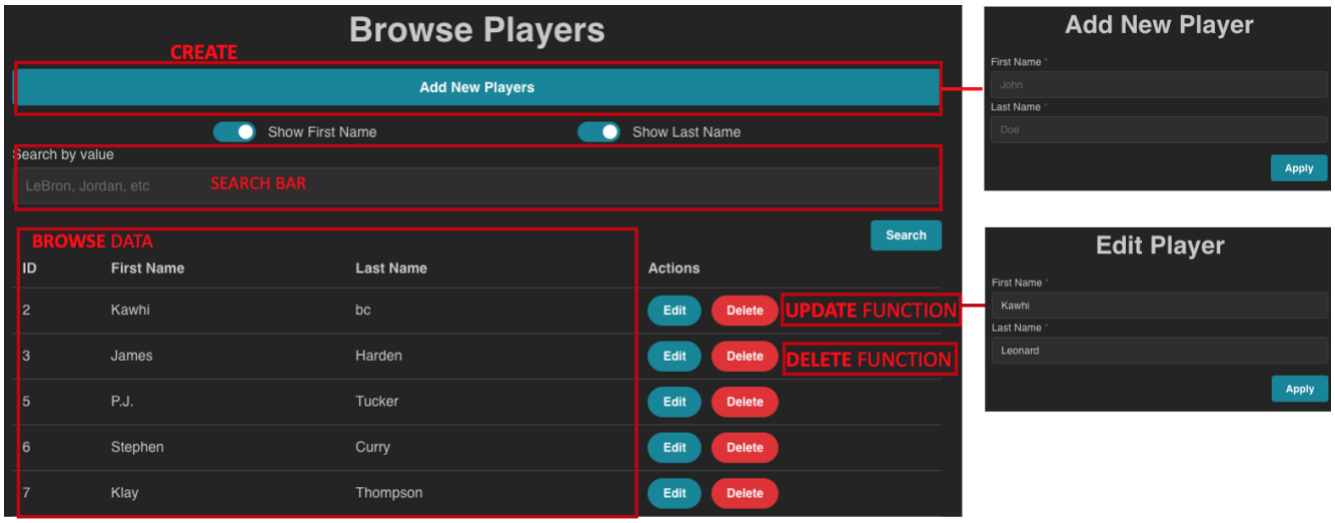


Figure 10: CRUD for Teams. CREATE/READ/UPDATE/DELETE Teams

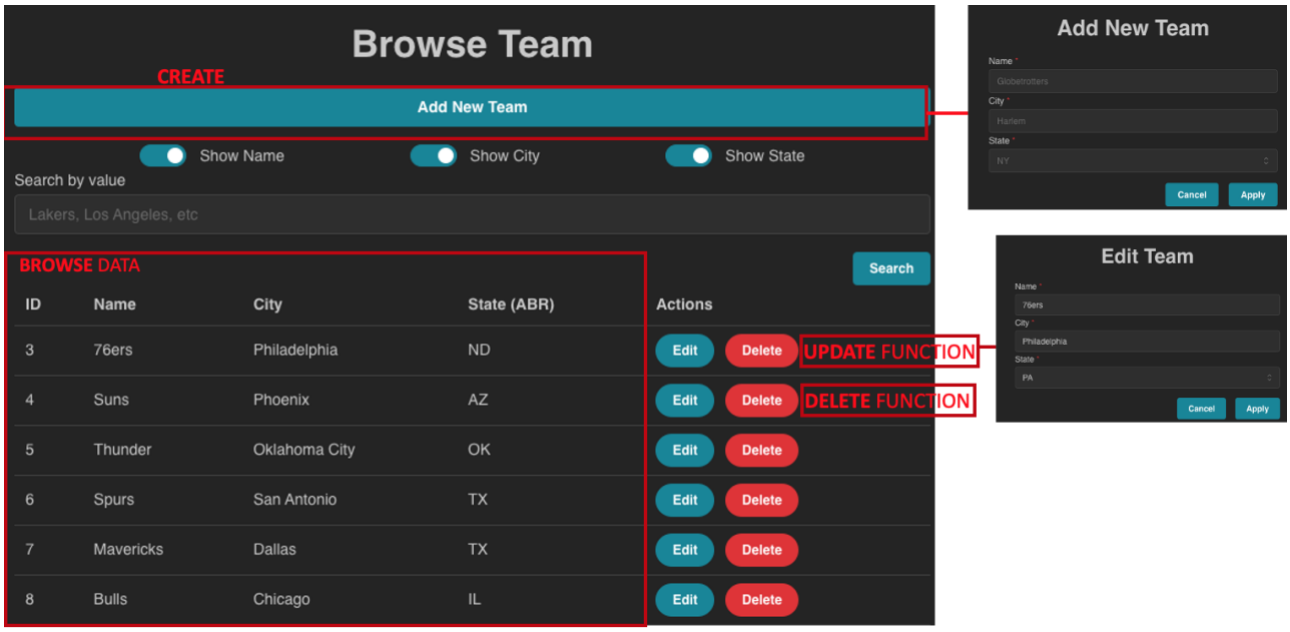
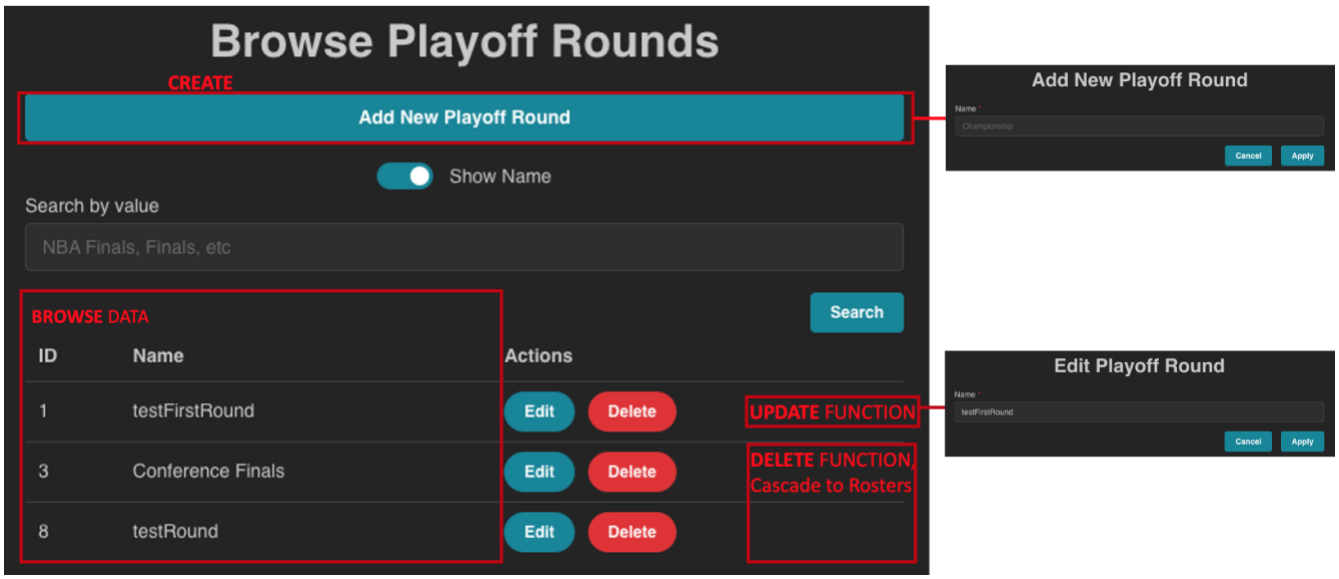


Figure 11:CRUD for Playoff Rounds. CREATE/READ/UPDATE/DELETE Playoff Rounds



Data Definition Queries

Please see enclosed document DDL.sql for full code.

```sql
INSERT INTO
    Teams (name, city, state)
VALUES
    ('Clippers', 'Los Angeles', 'CA'),
    ('Warriors', 'San Francisco', 'CA'),
    ('76ers', 'Philadelphia', 'PA'),
    ('Suns', 'Phoenix', 'AZ'),
    ('Thunder', 'Oklahoma City', 'OK'),
    ('Spurs', 'San Antonio', 'TX'),
    ('Mavericks', 'Dallas', 'TX'),
    ('Bulls', 'Chicago', 'IL'),
    ('Nets', 'Brooklyn', 'NY'),
    ('Wizards', 'Washington', 'DC'),
    ('Knicks', 'New York', 'NY'),
    ('Bucks', 'Milwaukee', 'WI'),
    ('Hawks', 'Atlanta', 'GA'),
    ('Lakers', 'Los Angeles', 'CA'),
    ('Celtics', 'Boston', 'MA');

INSERT INTO
    PlayoffRounds (name)
VALUES
    ('First Round'),
    ('Conference Semifinals'),
    ('Conference Finals'),
    ('NBA Finals');
```

Figure 12: Snippet of DDL.sql for this project.

Data Manipulation Queries

Please see enclosed document DML.sql for full code.

```sql
-- INSERT --
INSERT INTO PlayoffRounds (name) VALUES (:nameInput);

-- UPDATE --
UPDATE PlayoffRounds SET name=:nameInput WHERE playoffRoundID=:playoffRoundIDInput;

-- DELETE --
DELETE FROM PlayoffRounds WHERE playoffRoundID=:playoffRoundIDInput;

-- Rosters / RosterPlayers --
-- SELECT --
-- Full roster with associated team, playoff round, and players
SELECT
    r.rosterID,
    r.teamID,
    r.playoffRoundID,
    r.year,
    t.name AS teamName,
    t.city AS teamCity,
    t.state AS teamState,
    pr.name AS playoffRoundName,
    p.playerID AS playerID,
    p.firstName AS playerFirstName,
    p.lastName AS playerLastName,
FROM Rosters AS r
    JOIN Teams AS t ON r.teamID = t.teamID
    LEFT OUTER JOIN PlayoffRounds AS pr ON pr.playoffRoundID = r.playoffRoundID
    LEFT OUTER JOIN RosterPlayers AS rp ON rp.rosterID = r.rosterID
    LEFT OUTER JOIN Players AS p ON rp.playerID = p.playerID
ORDER BY r.year, t.name;
```

Figure 13: Snippet of DML.sql for this project.