

Project Proposal

Yong Hao (yh3290), Jerry Liu (jl6007)

We are planning to implement a Python application that operates the NBA database through a simple web front end. The database is designed to serve NBA fans, scouts, data analysts, etc., to help them compare players and analyze games throughout multiple seasons. Users can achieve the statistics of all NBA players and teams. Records can also be inquired by sorting certain domains, such as points, rebounds, assists, etc. The whole design includes seven entities: season, team, player, game, coach, player stats, team stats and contract. Details can be found in E-R diagram. The featured function of the application is to help users achieve data views in a more intuitive way – users can get the “radar chart” of a player, a team, or a coach that they inquiry. On top of that, users can choose specific perspectives to generate the radar chart they need and do the comparison.

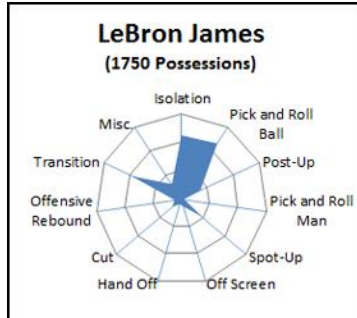
Data Source: we plan to combine these two Kaggle dataset and form our own database.

<https://www.kaggle.com/wyattowalsh/basketball>

<https://www.kaggle.com/nathanlauga/nba-games?select=ranking.csv>

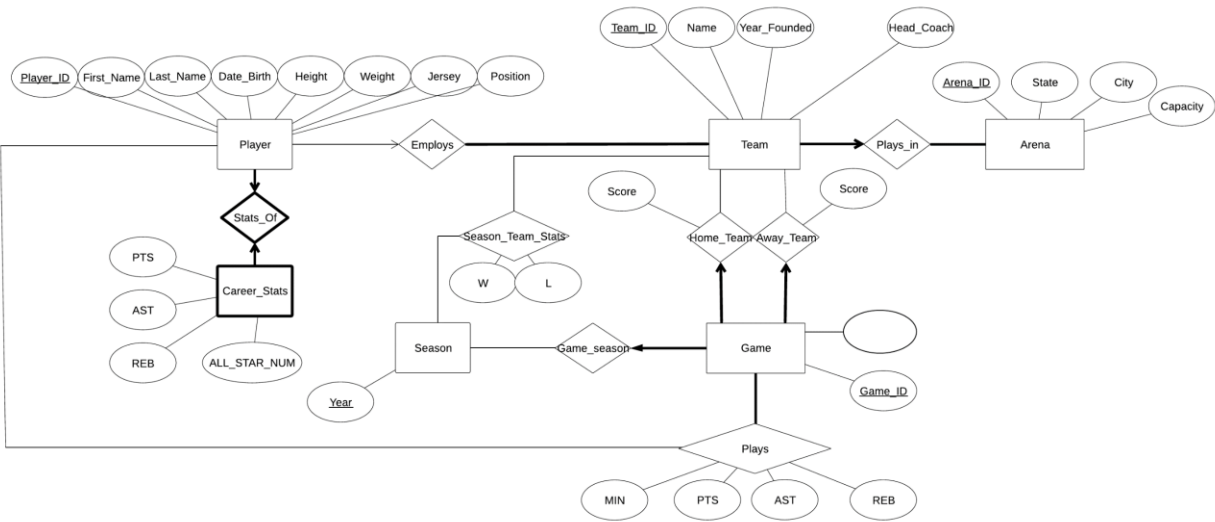
Part 3 Choice: A. Web Front-End Option

Radar Chart Sample:



Contingency Plan: If anyone of us dropped the class, the other one should finish the project but with fewer attributes per entity (for example, the entity *team* could delete *Year_Founded*, *Arena*, *City*, etc).

ER Diagram



Relational Schema

Project 1 Part 1 (For Part 2 see next page)

Player (Player_ID, First_Name, Last_Name, Date_Birth, Height, Weight, Jersey, Position, Team_ID NOT NULL,

PK(Player_ID), FK(Team_ID) -> Team)

Team (Team_ID, Name, Year_Founded, State, City, Arena,

PK(Team_ID))

Career_Stats (Player_ID, PTS, AST, REB, ALL_STAR_NUM,

PK(Player_ID) -> Player(Player_ID) ON DELETE CASCADE)

Season_Team_Stats (Year, Team ID, W, L,

PK(Year Team_ID), FK(Team_ID) -> Team)

Game (Year, Game_ID, Home_Team NOT NULL, Home_Team_Score NOT NULL, Home_Team_W/L NOT NULL,

Away_Team NOT NULL, Away_Team_Score NOT NULL, Away_Team_W/L NOT NULL,

PK(Game_ID), FK(Home_Team, Away_Team) -> Team)

Plays (Game_ID, Player_ID, MIN, PTS, AST, REB,

PK(Game_ID, Player_ID), FK(Game_ID) -> Game, FK(Player_ID) -> Player)

Relational Schema

Updated Oct 11 for Part 2

```
CREATE TABLE Player (  
    Player_ID INTEGER,  
    First_Name VARCHAR(15),  
    Last_Name VARCHAR(15),  
    Date_Birth DATE,  
    Height INTEGER,  
    Weight INTEGER,  
    Jersey INTEGER,  
    Position CHARACTER(2),  
    Team_ID INTEGER,  
    CAREER_PTS DECIMAL(2,1),  
    CAREER_AST DECIMAL(2,1),  
    CAREER_REB DECIMAL(2,1),  
    ALL_STAR_NUM INTEGER,  
    PRIMARY KEY (Player_ID),  
    FOREIGN KEY (Team_ID) REFERENCES Team  
);
```

```
CREATE TABLE Plays (  
    Game_ID INTEGER,  
    Player_ID INTEGER,  
    MIN INTEGER,  
    PTS INTEGER,  
    AST INTEGER,  
    REB INTEGER,  
    PRIMARY KEY (Player_ID, Game_ID),
```

```
FOREIGN KEY (Player_ID) REFERENCES Player,  
FOREIGN KEY (Game_ID) REFERENCES Game  
);
```

```
CREATE TABLE Game (  
    Game_ID INTEGER,  
    Home_Team_Win BOOLEAN,  
    Year INTEGER NOT NULL,  
    Home_Team_Score INTEGER NOT NULL,  
    Away_Team_Score INTEGER NOT NULL,  
    Home_Team_ID INTEGER NOT NULL,  
    Away_Team_ID INTEGER NOT NULL,  
    PRIMARY KEY (Game_ID),  
    FOREIGN KEY (Home_Team_ID) REFERENCES Team,  
    FOREIGN KEY (Away_Team_ID) REFERENCES Team,  
    CONSTRAINT Team_Check  
    CHECK (Home_Team_ID <> Away_Team_ID)  
);
```

```
CREATE TABLE Season_Team_Stats (  
    Team_ID INTEGER,  
    Year INTEGER,  
    W INTEGER,  
    L INTEGER,  
    PRIMARY KEY (Team_ID, Year),  
    FOREIGN KEY (Team_ID) REFERENCES Team  
);
```

```

CREATE TABLE Team (
    Team_ID INTEGER,
    Name VARCHAR(30),
    Year_Founded INTEGER,
    Head_Coach VARCHAR(30),
    Arena_ID INTEGER NOT NULL,
    PRIMARY KEY (Team_ID),
    FOREIGN KEY (Arena_ID) REFERENCES Arena
);

```

```

CREATE TABLE Arena (
    Arena_ID INTEGER,
    State CHARACTER(2),
    City VARCHAR(30),
    Capacity INTEGER,
    PRIMARY KEY (Arena_ID)
);

```

Below part is used to constrain ≥ 1 relations, yet doesn't supported by PostgreSQL:

```

CREATE ASSERTION Arena_Team
CHECK (
    NOT EXISTS (
        SELECT Arena_ID FROM Arena
        EXCEPT
        SELECT DISTINCT Arena_ID FROM Team)
);

```

```

CREATE ASSERTION Player_Team

```

```
CHECK (  
    NOT EXISTS (  
        SELECT Team_ID FROM Team  
    EXCEPT  
        SELECT DISTINCT Team_ID FROM Player)  
)
```

```
CREATE ASSERTION Player_Game
```

```
CHECK(  
    NOT EXISTS (  
        SELECT Game_ID FROM Game  
    EXCEPT  
        SELECT DISTINCT Game_ID FROM Plays)  
)
```