**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.

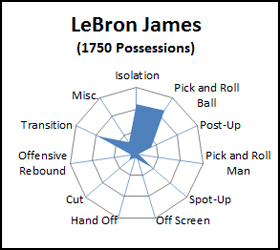
Data Source: we plan to combine these two Kaggle dataset and form our own database.

<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**

Diagram

Description automatically generated

**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(30),

Last\_Name VARCHAR(30),

Date\_Birth DATE,

Height INTEGER,

Weight INTEGER,

Jersey INTEGER,

Position CHARACTER(30),

Team\_ID INTEGER,

CAREER\_PTS DECIMAL(1,2),

CAREER\_AST DECIMAL(1,2),

CAREER\_REB DECIMAL(1,2),

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(30),

City VARCHAR(30),

Capacity INTEGER,

Areana\_Name VARCHAR(30),

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)

)

**Project 2**

CREATE TYPE Full\_Name AS (

First\_Name VARCHAR(30),

Last\_Name VARCHAR(30)

);

CREATE TABLE aNew\_Player (

Player\_ID INTEGER,

Name Full\_Name,

Date\_Birth DATE,

Height INTEGER,

Weight INTEGER,

Jersey INTEGER,

Position CHARACTER(30),

Team\_ID INTEGER,

CAREER\_STATS DECIMAL [],

ALL\_STAR\_NUM INTEGER,

PRIMARY KEY (Player\_ID),

FOREIGN KEY (Team\_ID) REFERENCES Team

);

CREATE TRIGGER check\_player\_del

BEFORE DELETE ON new\_player

FOR EACH ROW

EXECUTE FUNCTION if\_roll\_back();

CREATE OR REPLACE FUNCTION if\_roll\_back() RETURNS TRIGGER AS $example\_table$

BEGIN

IF (SELECT COUNT(\*) FROM new\_player WHERE team\_id = old.team\_id) <= 1 THEN

RETURN NULL;

ELSE

RETURN OLD;

END IF;

END;

$example\_table$ LANGUAGE plpgsql;