# CS275 Final Project

NBA Overview Database

By Anthony Clark

## OUTLINE: (5%)

Our database is concerned with tracking player and game information in the NBA. The purpose of our database would be to provide information to a casual fan who was searching out general facts regarding players and games recently played, or for someone who wanted to keep a running database to follow player movement, championships, and recent games.

With our NBA overview database, it is possible to add new players to teams, retire existing players, track championships won, view player positions, teams, and recent game scores, including the date they occurred and the points that each team scored. In addition, it gives everyone a chance to stare at the name Stephen Curry, which as we all know, is complete bliss.

## DATABASE OUTLINE WORDS: (5%)

### How the database is setup:

Our database is set up so that the team table (id, name, city) connects to both the player table, and the games table. The games table (id, team1Points, team2Points, tid1, tid2, gameDate) joins table teams on tid1 and tid2, which are both used as foreign keys. With these foreign keys, the names of each team involved in the game are provided. The player table (id, firstname, lastname, championships, tid) connects to the teams table on the foreign key tid, which provides each player entry with the team that they are currently on. From the players table exists a relationship to the playerPosition (id, plaid, posid) table, which acts as the conduit table for our many to many relationship. The playerPositions table joins the players table on the foreign key, posid, which gets the ID of the player whose position we're working with. The second piece of information stored with the player ID is the position ID (posid), which is a foreign key reference to the positions table, where the ID for each possible position is stored.

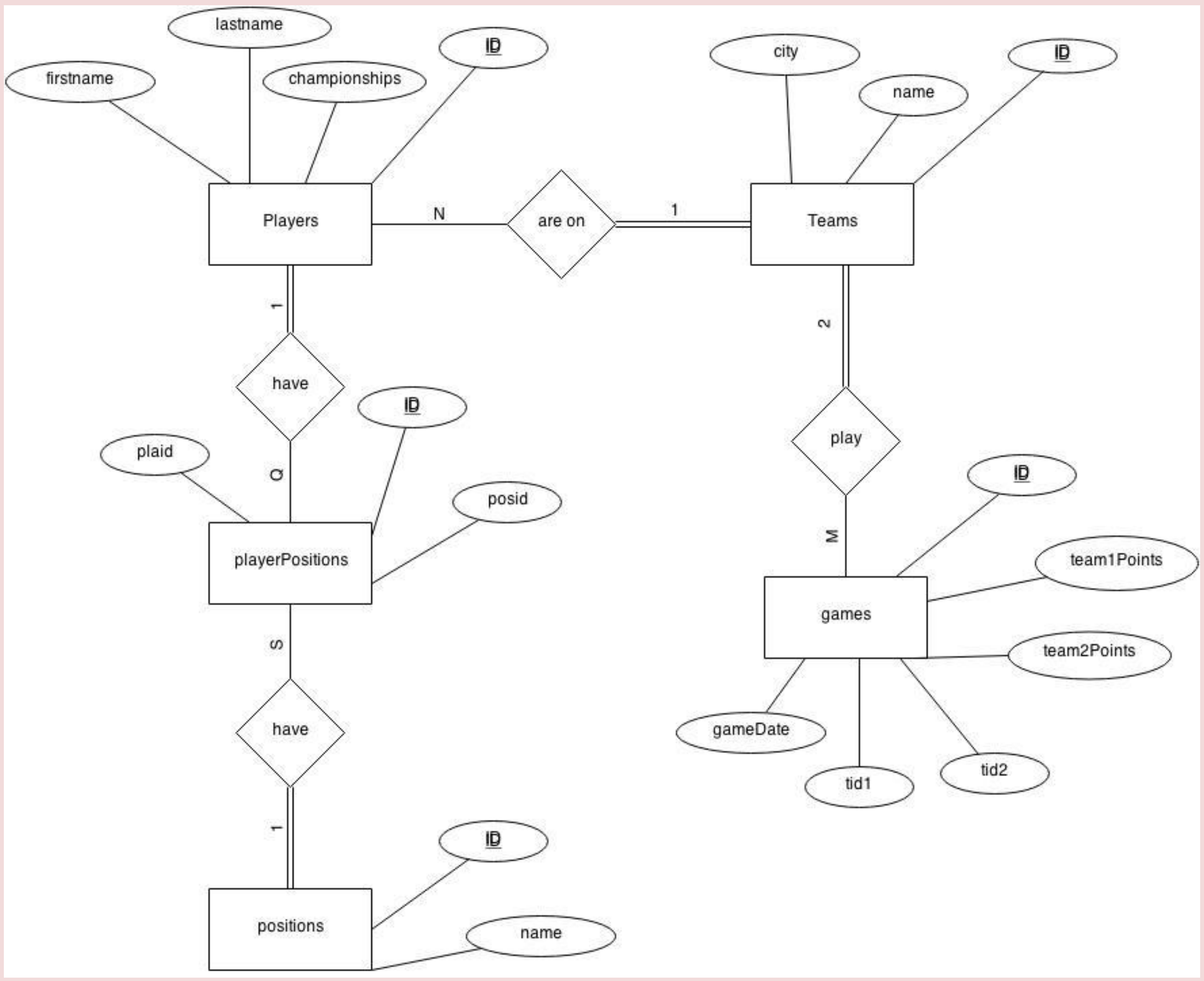
### Usage example: How to perform an insertion:

The first step in creating a new entry in the database is to add a team. Without a team in place, a player cannot be added, and without a player in place, a position cannot be added. Because the player table contains a foreign key (tid) that links up to the team name contained in the teams table, we've made it a requirement that no player can be added to a non-existent team. This was accomplished by creating a drop down box that is populated by the teams table, in order to constrict the amount of choices that the user has down to only teams that have already been entered.

Once a player has been created, they are then eligible to have a position added to their profile. This is accomplished through a many to many relationship, where a player may hold many positions (IE a player like LeBron who could be classified in multiple positions), and a position may hold many players. To create this relationship, a middle table called playerPositions was created. This table holds foreign key references to the players id (as plaid) and to their position id (as posid). Similar to the drop down for teams, a drop down for players was created to prevent the user from trying to assign a position to a player that doesn't exist.

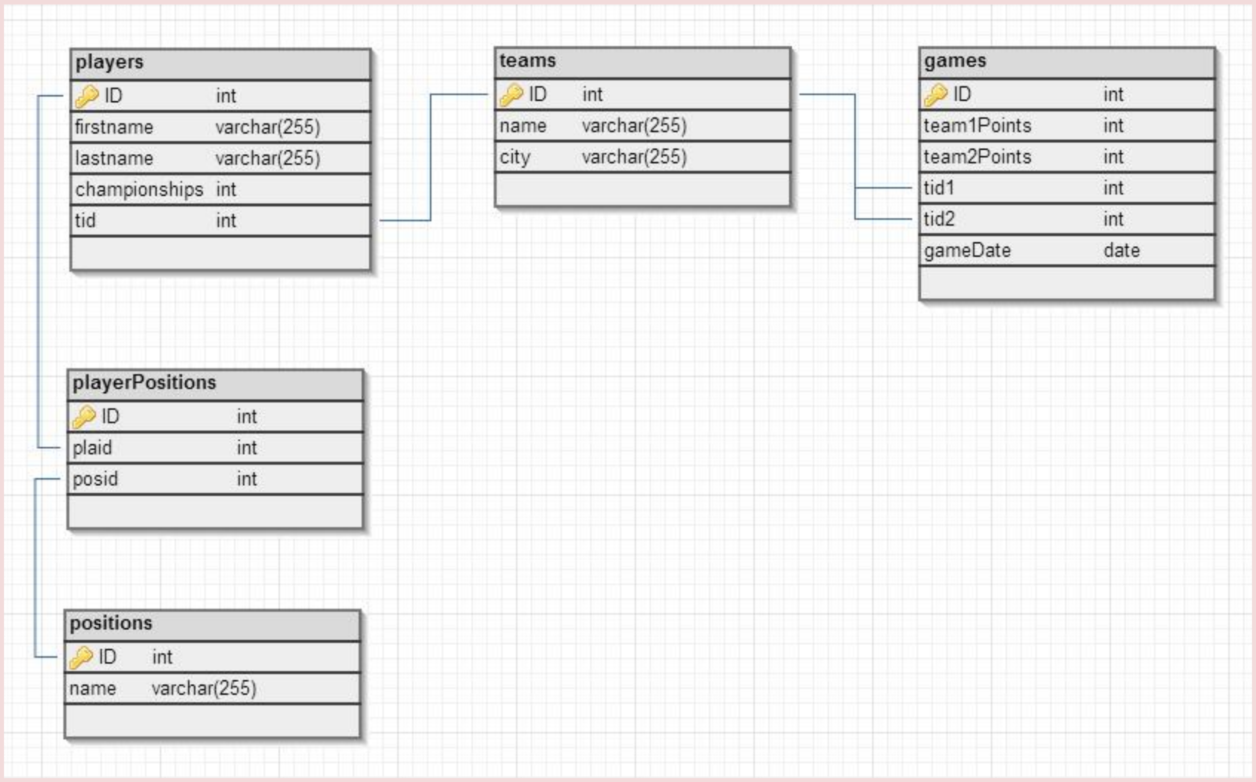
**Adding a new game requires the input of two scores, and an optional gameDate, which may be NULL. This is in the event a game is entered, but the date it was played is not remembered. The games table then pulls the id of team1 and team2 through a foreign key reference (tid1 and tid2) to team name in the team table.**

## ER DIAGRAM: (10%)



*playerPositions table is included for clarity purposes of Players table to positions table many to many relationship only (would not normally be shown in ER diagram).

## DATABASE SCHEMA: (10%)



## TABLE CREATION QUESTIES: (%10)

**Create Table players:**

```
CREATE TABLE players (
        ID int NOT NULL AUTO_INCREMENT,
        firstname VARCHAR(255) NOT NULL,
        lastname VARCHAR(255) NOT NULL,
        championships int,
        tid int NOT NULL,
        PRIMARY KEY (ID),
        FOREIGN KEY (tid) REFERENCES teams (ID) ON DELETE RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

**Create Table teams:**

```
CREATE TABLE teams (
        ID int NOT NULL AUTO_INCREMENT,
        name VARCHAR(255) NOT NULL,
        city VARCHAR(255) NOT NULL,
        PRIMARY KEY (ID),
        KEY (name)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

**Create Table positions:**

```
CREATE TABLE positions (
		ID int NOT NULL AUTO_INCREMENT,
		name VARCHAR(255) NOT NULL,
		PRIMARY KEY (ID),
		KEY (name)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

**Create Table playerPositions**

```
CREATE TABLE playerPositions (
		ID int NOT NULL AUTO_INCREMENT,
		plaid int NOT NULL,
		posid int NOT NULL,
		PRIMARY KEY (ID),
		KEY (plaid),
		KEY (posid),
		FOREIGN KEY (plaid) REFERENCES players (ID) ON DELETE RESTRICT ON UPDATE CASCADE,
		FOREIGN KEY (posid) REFERENCES positions (ID) ON DELETE RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

**Create Table games**

```
CREATE TABLE games (
		ID int NOT NULL AUTO_INCREMENT,
		team1Points INT NOT NULL,
		team2Points INT NOT NULL,
		tid1 INT NOT NULL,
		tid2 INT NOT NULL,
		gameDate DATE,
		PRIMARY KEY (ID),
		KEY (tid1),
		KEY (tid2),
		FOREIGN KEY (tid1) REFERENCES teams (ID) ON DELETE RESTRICT ON UPDATE CASCADE,
		FOREIGN KEY (tid2) REFERENCES teams (ID) ON DELETE RESTRICT ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

## GENERAL USE QUERIES: (%30)

**Add a team:**

```
INSERT INTO teams(name, city)
VALUES([teamName], [teamCity])";
```

**Select a team:**

```
SELECT teams.name, teams.city
FROM teams
```

**Add a player:**

```
INSERT INTO players(tid, firstname, lastname, championships)
VALUES((SELECT ID FROM teams WHERE name = [teamName]), [firstName], [lastName],
[championships])
```

**Select a player:**

```
SELECT players.firstname, players.lastname, players.championships, teams.name, teams.city
FROM players
INNER JOIN teams
```

```
ON teams.ID = players.tid
```

**Add a position:**

```
INSERT INTO playerPositions(plaid, posid)
VALUES((SELECT ID FROM players WHERE lastname = [lastName]),
(SELECT ID FROM positions WHERE name = [position]))
```

**Select a position:**

```
SELECT players.firstname, players.lastname, positions.name
FROM players
INNER JOIN playerPositions
ON players.ID = playerPositions.plaid
INNER JOIN positions
ON playerPositions.posid = positions.ID
```

**Add a game:**

```
INSERT INTO games(team1Points, team2Points, tid1, tid2, gameDate)
VALUES([team1Score], [team2Score], (SELECT ID FROM teams WHERE name = [team1]),
(SELECT ID FROM teams WHERE name = [team2]), [gameDate]
```

**Select a game:**

```
SELECT teams1.name, teams2.name AS name2, games.team1Points, games.team2Points, gameDate
FROM games
INNER JOIN teams AS teams1
ON teams1.ID = games.tid1
INNER JOIN teams AS teams2
ON teams2.ID = games.tid2
ORDER BY gameDate
```

**Update championships: (No selection - displayed in player selection)**

```
UPDATE players
SET championships = [championships]
WHERE firstname = [firstName] AND [lastname] = [lastName]
```

**Delete a player (requires a position delete from many to many table playerPosition)**

```
/*Get player ID*/
PlayerID = SELECT ID FROM players WHERE firstname = [firstName] AND lastname = [lastName]
/*Get position ID*/
PositionID = SELECT ID FROM positions WHERE name = [position]
/*Delete Player From playerPosition Table*/
DELETE FROM playerPositions
WHERE plaid = [PlayerID] AND posid = [PositionID]
/*Delete Player From players  Table*/
DELETE
FROM players
WHERE ID = [PlayerID]
```

**Select all player info:**

```
SELECT players.firstname, players.lastname, players.championships, positions.name AS nameP,
teams.name, teams.city
FROM players
INNER JOIN teams
ON teams.ID = players.tid
```

```
INNER JOIN playerPositions
ON players.ID = playerPositions.plaid
INNER JOIN positions
ON playerPositions.posid = positions.ID
ORDER BY players.lastname
```

**Select player info by team:**

```
SELECT players.firstname, players.lastname, players.championships, positions.name AS nameP,
teams.name, teams.city
FROM players
INNER JOIN teams
ON teams.ID = players.tid
INNER JOIN playerPositions
ON players.ID = playerPositions.plaid
INNER JOIN positions
ON playerPositions.posid = positions.ID
WHERE teams.name = [teamName]
ORDER BY players.lastname
```

**Select game info by team:**

```
SELECT teams1.name, teams2.name AS name2, games.team1Points, games.team2Points, gameDate
FROM games
INNER JOIN teams AS teams1
ON teams1.ID = games.tid1
INNER JOIN teams AS teams2
ON teams2.ID = games.tid2
WHERE teams1.name = [teamName] OR teams2.name = [teamName]
ORDER BY gameDate
```