NBA Database

Brian Tracey
Professor Labouseur
Database Management
26 April 2016

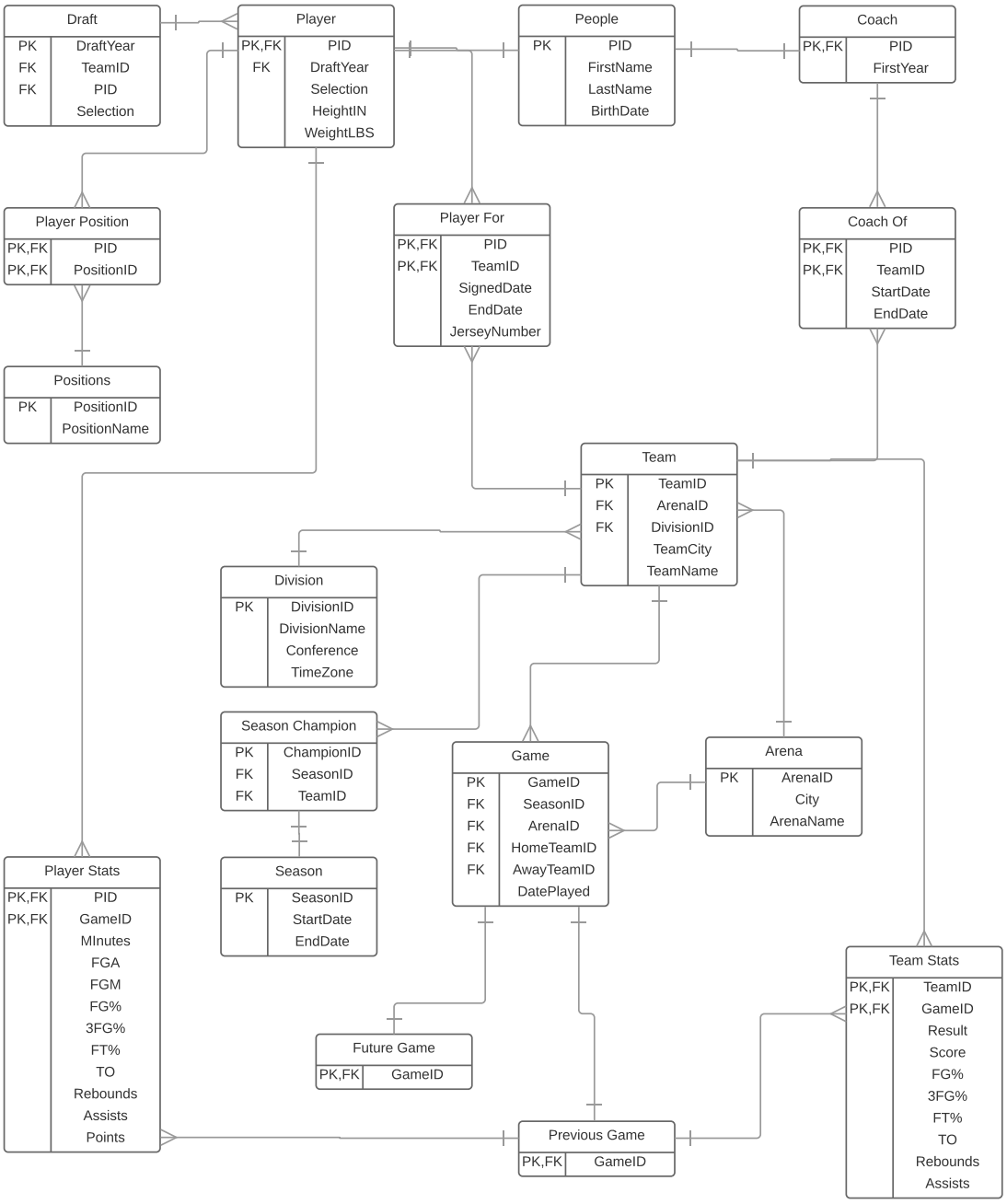
Table of Contents

Executive Summary

This database keeps a record of all former and current players and teams in the National Basketball Association. It holds each player's stats, which teams each player has played for and when, and a player's draft selection and year. It also keeps track of teams wins and losses, game stats and rosters and coaches for each season.

This database is to be used by fans that want to keep a log of his or her team or favorite players. The following pages include an entity relationship diagram, information about the tables within the database, security permissions and examples of certain stored procedures, views, reports and triggers.

Entity Relationship Diagram



Tables

People Table

The People Table holds basic information such as first and last name and birth date. This table will be expanded into coaches and players.

Functional Dependencies

PID -> FirstName, LastName, BirthDate

Create Statement

```
--People Table

CREATE TABLE People (
        PID             INTEGER         NOT NULL,
        FirstName       TEXT            NOT NULL,
        LastName        TEXT            NOT NULL,
        BirthDate       DATE            NOT NULL,
        PRIMARY KEY (PID)
        );
```

People Output

```
SELECT * FROM people
```

Output pane

Data Output | Explain | Messages | History

	pid integer	firstname text	lastname text	birthdate date
1	1	Isaiah	Thomas	1989-02-07
2	2	Avery	Bradley	1990-11-26
3	3	Jae	Crowder	1990-04-06
4	4	Evan	Turner	1988-10-27
5	5	Jared	Sullinger	1992-03-04
6	6	Brad	Stevens	1976-10-22
7	7	Shane	Larkin	1992-10-02
8	8	Brooke	Lopez	1988-04-01
9	9	Thaddeus	Young	1988-06-21
10	10	Jarret	Jack	1983-10-28
11	11	Bojan	Bogdanovic	1989-04-18
12	12	Tony	Brown	1960-07-29
13	13	Carmelo	Anthony	1984-05-29
14	14	Kristaps	Porzingis	1995-08-02
15	15	Tony	Wroten	1993-04-13
16	16	Derrick	Williams	1991-05-25
17	17	Aaron	Afflalo	1985-10-15
18	18	Kurt	Rambis	1958-02-25
19	19	Isaiah	Canaan	1991-05-21
20	20	Ish	Smith	1988-07-05

Player Table

The player table is a sub table of the people table. It specifies which people are players and includes his draft year, selection, height and weight.

Functional Dependencies

PID -> DraftYear, Selection, HeightIN, WeightLBS

Create Statement

```
CREATE TABLE Player (
        PID             INTEGER         NOT NULL REFERENCES People (PID),
        DraftYear       SMALLINT        NOT NULL REFERENCES Draft (DraftYear),
        Selection       TEXT            NOT NULL DEFAULT 'Undrafted',
        HeightIN        SMALLINT        NOT NULL CHECK (HeightIN > 0),
        WeightLBS       SMALLINT        NOT NULL CHECK (WeightLBS > 0),
        PRIMARY KEY (PID)
        );
```

Player Output

Previous queries

```
SELECT * FROM player
```

Output pane

Data Output | Explain | Messages | History

	pid integer	draftyear smallint	selection text	heightin smallint	weightlbs smallint
1	1	2002	7	70	170
2	33	2004	7	75	185
3	14	2003	4	85	230
4	37	2001	3	72	205
5	28	2002	5	83	265
6	22	2004	8	82	245
7	34	2003	3	78	225
8	11	2001	17	81	235
9	27	2001		75	205

Player_For Table

The Player_For table shows which player has player for what teams and when he was signed and when the contract (with that team) ended. It also includes his jersey numbers.

Functional Dependencies

PID, TeamID -> SignedDate, EndDate, JerseyNumber

Create Statement

```
CREATE TABLE Player_For (
        PID             INTEGER         NOT NULL REFERENCES People (PID),
        TeamID          INTEGER         NOT NULL REFERENCES Team (TeamID),
        SignedDate      DATE            NOT NULL,
        EndDate         DATE            NOT NULL,
        JerseyNumber    INTEGER         NOT NULL CHECK (JerseyNumber >= 0),
        PRIMARY KEY (PID,TeamID)
        );
```

Player For Output

```
SELECT * FROM PLayer_for
```

utput pane

Data Output | Explain | Messages | History

	pid integer	teamid integer	signeddate date	enddate date	jerseynumber integer
1	1	22	2014-01-06	2016-06-20	4
2	2	22	2012-07-20	2016-06-20	0
3	33	13	2009-07-20	2016-06-20	30
4	40	20	2009-07-20	2016-06-20	32
5	22	30	2015-07-20	2016-06-20	8
6	29	1	2012-07-20	2016-06-20	30
7	3	22	2014-01-06	2016-06-20	99
8	4	22	2014-01-06	2016-06-20	10
9	5	22	2014-01-06	2016-06-20	8

Player Position Table

The Players_Position Table is a weak entity table the links a player to his position.

Functional Dependencies

None

Create Statement

```
CREATE TABLE Player_Position (
        PID             INTEGER         NOT NULL REFERENCES People (PID),
        PositionID      INTEGER         NOT NULL REFERENCES Positions (PositionID),
        PRIMARY KEY (PID, PositionID)
        );
```

Player_Position Output

```
SELECT * FROM PLayer_position
```

tput pane

Data Output | Explain | Messages | H

	pid integer	positionid integer
1	1	1
2	2	2
3	3	3
4	4	3
5	5	4
6	8	5

Position Table

The Position Table lists out the five positions a player can be; Point Guard, Shooting Guard, Small Forward, Power Forward, Center.

Functional Dependencies

PositionID-> PositionName

Create Statement

```
--Positions Table

CREATE TABLE Positions (
        PositionID      INTEGER         NOT NULL,
        PositionName    TEXT            NOT NULL,
        PRIMARY KEY (positionID)
        );
```

Positions Output

```
SELECT * FROM positions
```

utput pane

Data Output | Explain | Message:

	positionid integer	positionname text
1	1	Point Guard
2	2	Shooting Guard
3	3	Small Forward
4	4	Power Forward
5	5	Center

Draft Table

The Draft table includes the draft year as the primary key and will show the first selection and the team that selected him for that draft year.

Functional Dependencies

DraftYear -> TeamID, PID, Selection

Create Statement

```
CREATE TABLE Draft (
        DraftYear       SMALLINT        NOT NULL,
        TeamID          INTEGER         NOT NULL REFERENCES Team (TeamID),
        PID             INTEGER         NOT NULL REFERENCES People (PID),
        Selection       SMALLINT        NOT NULL CHECK (Selection = 1),
        PRIMARY KEY (DraftYear)
        );
```

Draft Output

```
SELECT * FROM draft
```

utput pane

Data Output | Explain | Messages

	draftyear smallint	teamid integer	pid integer	selection smallint
1	2000	7	33	1
2	2001	27	1	1
3	2002	25	15	1
4	2003	16	37	1
5	2004	4	16	1
6	2005	9	19	1

Coach Table

The Coach Table is an extension of the People table but for coaches.

Functional Dependencies

PID -> FirstYear

Create Statement

```
--Coach Table

CREATE TABLE Coach (
        PID             INTEGER         NOT NULL REFERENCES People (PID),
        FirstYear       DATE            NOT NULL,
        PRIMARY KEY (PID)
        );
```

Coach Output

```
SELECT * FROM coach
```

tput pane

Data Output | Explain | Me

	pid integer	firstyear integer
1	6	2013
2	12	2016
3	18	2016
4	24	2013
5	30	2011
6	36	2014
7	42	2013

Coach Of Table

The Coach Of table links the coach to the team he coaches along with his start and end date.

Functional Dependencies

PID , TeamID -> StartDate, EndDate

Create Statement

```
CREATE TABLE Coach_Of (
        PID             INTEGER         NOT NULL REFERENCES People (PID),
        TeamID          INTEGER         NOT NULL REFERENCES Team (TeamID),
        StartDate       DATE            NOT NULL,
        EndDate         DATE            NOT NULL,
        PRIMARY KEY (PID)
        );
```

Coach Of Output

Previous queries

```
SELECT * FROM coach_of
```

Output pane

Data Output | Explain | Messages | History

	pid integer	teamid integer	startdate date	enddate date
1	6	23	2014-07-04	2016-06-20
2	12	7	2016-02-08	2016-04-15
3	18	11	2016-01-26	2016-06-20
4	24	29	2013-06-14	2016-06-20
5	30	1	2011-06-20	2016-06-20
6	36	13	2014-06-14	2016-06-20

Arena Table

The Arena Table shows what arena teams play in. This can show fans of the game, which teams play well in what arenas.

Functional Dependencies

ArenaID -> City, Name

Create Statement

```
CREATE TABLE Arena (
        ArenaID         INTEGER         NOT NULL,
        City            TEXT            NOT NULL,
        ArenaName       TEXT            NOT NULL,
        PRIMARY KEY (ArenaID)
        );
```

Arena Output

```
SELECT * FROM Arena
```

Output pane

Data Output | Explain | Messages | History

	arenaid integer	city text	arenaname text
1	1	Toronto	Air Canada Centre
2	2	Miami	American Airlines Arena
3	3	Dallas	American Airlines Center
4	4	Orlando	Amway Center
5	5	San Antonio	AT&T Center
6	6	Indianopolis	Bankers Life Fieldhouse
7	7	New York	Barclays Center
8	8	Milwaukee	BMO Harris Bradley Center
9	9	Oklahoma City	Chesapeake Energy Arena
10	10	Memphis	FedExForum
11	11	New York City	Madison Square Garden
12	12	Portland	Moda Center
13	13	Oakland	Oracle Arena
14	14	Denver	Pepsi Center
15	15	Atlanta	Philips Arena
16	16	Cleveland	Quicken Loans Arena
17	17	Sacramento	Sleep Train Arena
18	18	New Orleans	Smoothie King Center

Division Table

The Division Table shows which teams are in which division and which conference. This can be useful if users of the database want to see the standings and who will make playoffs.

Functional Dependencies

DivisionID -> DivisionName, Conference, TimeZone

Create Statement

```
CREATE TABLE Division (
        DivisionID      INTEGER         NOT NULL,
        DivisionName    TEXT            NOT NULL,
        Conference      TEXT            NOT NULL,
        TimeZone        TEXT            NOT NULL DEFAULT 'Eastern',
        PRIMARY KEY (DivisionID)
        );
```

Division Output

```
SELECT * FROM division
```

utput pane

Data Output | Explain | Messages | History

	divisionid integer	divisionname text	conference text	timezone text
1	1	Atlantic	Eastern	Eastern
2	2	Southeast	Eastern	Eastern
3	3	Central	Eastern	Central
4	4	Pacific	Western	Pacific
5	5	Southwest	Western	Central
6	6	Northwest	Western	Mountain West

Team Table

The team table lists each team in the NBA. This is necessary to show the rosters for each team, who coaches each team and then stats and results of games. This is one of the more primary tables in the database.

Functional Dependencies

TeamID -> ArenaID, DivisionID, TeamCity, TeamName

Create Statement

```
CREATE TABLE Team (
        TeamID          INTEGER         NOT NULL,
        ArenaID         INTEGER         NOT NULL REFERENCES Arena (ArenaID),
        DivisionID      INTEGER         NOT NULL REFERENCES Division (DivisionID),
        TeamCity        TEXT            NOT NULL,
        TeamName        TEXT            NOT NULL,
        PRIMARY KEY (TeamID)
        );
```

Team Output

```
SELECT * FROM team
```

utput pane

Data Output | Explain | Messages | History

	teamid integer	arenaid integer	divisionid integer	teamcity text	teamname text
1	1	1	1	Toronto	Raptors
2	2	2	2	Miami	Heat
3	3	3	5	Dallas	Mavericks
4	4	4	2	Orlando	Magic
5	5	5	5	San Antonio	Spurs
6	6	6	3	Indiana	Pacers
7	7	7	1	Brooklyn	Nets
8	8	8	3	Milwaukee	Bucks
9	9	9	6	Oklahoma City	Thunder
10	10	10	5	Memphis	Grizzlies
11	11	11	1	New York	Knicks
12	12	12	6	Portland	Trailblazers
13	13	13	4	Golden State	Warriors
14	14	14	6	Denver	Nuggets
15	15	15	2	Atlanta	Hawks
16	16	16	3	Cleveland	Cavaliers
17	17	17	4	Sacramento	Kings
18	18	18	5	New Orleans	Pelicans

Season Champion Table

This table is necessary to show the NBA Champion for each season.

Functional Dependencies

ChampionID -> SeasonID, TeamID

Create Statement

```
CREATE TABLE Season_Champion (
        ChampionID      INTEGER         NOT NULL,
        SeasonID        INTEGER         NOT NULL REFERENCES Season (SeasonID),
        TeamID          INTEGER         NOT NULL REFERENCES Team (TeamID),
        PRIMARY KEY (ChampionID)
        );
```

Season Champion Output

```
SELECT * FROM Season_champion
```

utput pane

Data Output | Explain | Messages

	championid integer	seasonid integer	teamid integer
1	1	1	19
2	2	2	3
3	3	3	2
4	4	4	2
5	5	5	5
6	6	6	13

Season Table

The Season Table lists out each season's start date and end date. This is necessary to show which team won a championship for that season.

Functional Dependencies

SeasonID -> StartDate, EndDate

Create Statement

```
CREATE TABLE Season(
        SeasonID        INTEGER         NOT NULL,
        StartDate       DATE            NOT NULL,
        EndDate         DATE            NOT NULL,
        PRIMARY KEY (SeasonID)
        );
```

Season Output

```
SELECT * FROM Season
```

tput pane

Data Output | Explain | Messages

	seasonid integer	startdate date	enddate date
1	1	2009-09-01	2010-06-20
2	2	2010-09-01	2011-06-20
3	3	2011-09-01	2012-06-20
4	4	2012-09-01	2013-06-20
5	5	2013-09-01	2014-06-20
6	6	2014-09-01	2015-06-20
7	7	2015-09-01	2016-06-20

Game Table

The Game Table lists out the games played in each season. It has the two teams playing, one as home team and one as away team and which arena it is played in. it also includes the date. The games table gets broken down into a previous games table and then a future games table.

Functional Dependencies

GameID -> SeasonID, ArenaID, HomeTeamID, AwayTeamID, DatePlayed

Create Statement

```
--Game Table

CREATE TABLE Game (
        GameID          INTEGER         NOT NULL,
        SeasonID        INTEGER         NOT NULL REFERENCES Season (SeasonID),
        ArenaID         INTEGER         NOT NULL REFERENCES Arena (ArenaID),
        HomeTeamID      INTEGER         NOT NULL REFERENCES Team (TeamID),
        AwayTeamID      INTEGER         NOT NULL REFERENCES Team (TeamID),
        DatePlayed      DATE            NOT NULL,
        PRIMARY KEY (GameID)
        );
```

Game Output

```
SELECT * FROM Game
```

tput pane

Data Output | Explain | Messages | History

	gameid integer	seasonid integer	arenaid integer	hometeamid integer	awayteamid integer	dateplayed date
1	1	7	1	1	22	2016-03-15
2	2	7	13	13	20	2016-02-19
3	3	7	7	7	11	2016-01-12
4	4	7	2	2	16	2015-12-25
5	5	7	5	5	9	2016-03-01

Previous Game Table

This table shows that a game has already been played. It is an extension of the Game table. It then links this game to include stats for both players and teams.

Functional Dependencies

None

Create Statement

```
CREATE TABLE Previous_Game (
        GameID          INTEGER         NOT NULL REFERENCES Game (GameID),
        PRIMARY KEY (GameID)
        );
```

Previous Game Output

```
SELECT * FROM Previous_Game
```

utput pane

Data Output | Explain | Messages

	gameid integer
1	1
2	2
3	3

Future Game Table

This table shows that a will be played in the future. It is an extension of the Game table.

Functional Dependencies

None

Create Statement

```
CREATE TABLE Future_Game (
        GameID          INTEGER         NOT NULL REFERENCES Game (GameID),
        PRIMARY KEY (GameID)
        );
```

Future Game Output

```
SELECT * FROM Future_Game
```

tput pane

Data Output | Explain | Messag

	gameid integer
1	4
2	5

Player Stats Table

This table shows the stats each player has recorded in games he has played.

Functional Dependencies

PID, GameID -> Minutes, FGA, FGM, FGPercent, ThreePointPercent,
FreeThrowPercent, TurnOver, Assists, Rebounds, Points

Create Statement

```
CREATE TABLE Player_Stats (
        PID                     INTEGER         NOT NULL REFERENCES People (PID),
        GameID                  INTEGER         NOT NULL REFERENCES Game (GameID),
        Minutes                 INTEGER         NOT NULL CHECK (Minutes >= 0),
        FGA                     INTEGER         NOT NULL CHECK (FGA >=0),
        FGM                     INTEGER         NOT NULL CHECK (FGM >=0),
        FGPercent               SMALLINT        NOT NULL CHECK (FGPercent >= 0 AND FGPercent <= 100),
        ThreePointPercent       SMALLINT        NOT NULL CHECK (ThreePointPercent >= 0 AND ThreePointPercent <= 100),
        FreeThrowPercent        SMALLINT        NOT NULL CHECK (FreeThrowPercent >= 0 AND FreeThrowPercent <= 100),
        TurnOver                INTEGER         NOT NULL CHECK (TurnOver >=0),
        Assists                 INTEGER         NOT NULL CHECK (Assists >=0),
        Rebounds                INTEGER         NOT NULL CHECK (Rebounds >=0),
        Points                  INTEGER         NOT NULL CHECK (Points >=0),
        PRIMARY KEY (PID, GameID)
        );
```

Player Stats Output

```
SELECT * FROM player_stats
```

utput pane

Data Output | Explain | Messages | History

	pid integer	gameid integer	minutes integer	fga integer	fgm integer	fgpercent smallint	threepointpercent smallint	freethrowpercent smallint	turnover integer	assists integer	rebounds integer	points integer
1	1	1	36	19	11	58	34	75	4	6	4	29
2	27	1	35	15	5	33	30	40	3	5	5	13
3	33	2	38	22	13	59	50	75	4	7	4	42
4	35	2	38	13	8	62	0	75	4	10	9	23
5	37	2	40	20	10	50	33	100	7	16	5	22
6	40	2	33	18	7	39	50	90	6	6	9	23

Team Stats Table

This table shows the stats a team had in each game they have played.

Functional Dependencies

TeamID, GameID -> Result, Score, FGPercent, ThreePointPercent,
FreeThrowPercent, TurnOver, Assists, Rebounds

Create Statement

```
--Team Stats Table
CREATE TYPE Result AS ENUM ('W','L');


CREATE TABLE Team_Stats (
        TeamID                  INTEGER         NOT NULL REFERENCES Team (TeamID),
        GameID                  INTEGER         NOT NULL REFERENCES Game (GameID),
        Result                  TEXT            NOT NULL,
        Score                   TEXT            NOT NULL,
        FGPercent               SMALLINT        NOT NULL CHECK (FGPercent >= 0 AND FGPercent <= 100),
        ThreePointPercent       SMALLINT        NOT NULL CHECK (ThreePointPercent >= 0 AND ThreePointPercent <= 100),
        FreeThrowPercent        SMALLINT        NOT NULL CHECK (FreeThrowPercent >= 0 AND FreeThrowPercent <= 100),
        Turnover                INTEGER         NOT NULL CHECK (Turnover >=0),
        Assists                 INTEGER         NOT NULL CHECK (Assists >=0),
        Rebounds                INTEGER         NOT NULL CHECK (Rebounds >=0),
        PRIMARY KEY (TeamID, GameID)
        );
```

Team Stats Output

```
SELECT * FROM Team_stats
```

utput pane

Data Output | Explain | Messages | History

	teamid integer	gameid integer	result text	score text	fgpercent smallint	threepointpercent smallint	freethrowpercent smallint	turnover integer	assists integer	rebounds integer
1	1	1	L	109-106	42	34	77	22	21	29
2	22	1	W	109-106	44	39	82	14	24	33
3	13	2	W	122-116	51	41	83	16	29	27
4	20	2	L	122-116	49	37	76	20	28	33

Reports

First Report

The First report will show the years within one season.

Statement

```
SELECT EXTRACT(YEAR FROM Season.StartDate) AS "Start Year", EXTRACT(YEAR FROM Season.EndDate) AS "End Year"
FROM Season
WHERE now() > Season.EndDate;
```

Output

	Start Year double precision	End Year double precision
1	2009	2010
2	2010	2011
3	2011	2012
4	2012	2013
5	2013	2014
6	2014	2015

Second Report

The second report returns the list of coaches in the league, their date of birth and the team they coach.

Statement

```
SELECT people.FirstName, people.LastName, people.BirthDate, team.teamname
FROM people
INNER JOIN coach_of
ON people.PID=coach_of.pid
INNER JOIN team ON coach_of.teamID = team.TeamID;
```

Output

	firstname text	lastname text	birthdate date	teamname text
1	Brad	Stevens	1976-10-22	Pistons
2	Tony	Brown	1960-07-29	Nets
3	Kurt	Rambis	1958-02-25	Knicks
4	Brett	Brown	1961-02-16	76ers
5	Dwane	Casey	1957-04-17	Raptors
6	Steve	Kerr	1965-09-27	Warriors

Third Report

The third report lists players in the league and their current team.

Statement

```
SELECT people.FirstName, people.LastName, people.BirthDate, team.teamname
FROM people
INNER JOIN player_for
ON people.PID=player_for.pid
INNER JOIN team ON player_for.teamID = team.TeamID;
```

Output

	firstname text	lastname text	birthdate date	teamname text
1	Isaiah	Thomas	1989-02-07	Celtics
2	Avery	Bradley	1990-11-26	Celtics
3	Stephen	Curry	1988-03-14	Warriors
4	Blake	Griffin	1989-03-16	Clippers
5	Jahlil	Okafor	1995-12-15	Suns
6	Terrence	Ross	1991-02-05	Raptors

Views

First View

The first view statement allows the user to see a Team's Arena name, sorted by alphabetical order.

Statement

```
CREATE OR REPLACE VIEW TeamArenas AS
SELECT Team.TeamName AS Team_Name,
Team.TeamCity AS Team_City,
Arena.ArenaName AS Arena_Name
FROM Team, Arena

WHERE team.arenaID=Arena.arenaID
GROUP BY team.teamname, team.teamcity, arena.arenaname
ORDER BY team.teamcity;
```

Output

```
SELECT * FROM TeamArenas
```

put pane

ata Output | Explain | Messages | History

	team_name text	team_city text	arena_name text
1	Hawks	Atlanta	Philips Arena
2	Celtics	Boston	TD Garden
3	Nets	Brooklyn	Barclays Center
4	Hornets	Charlotte	Time Warner Cable Arena
5	Bulls	Chicago	United Center
6	Cavaliers	Cleveland	Quicken Loans Arena
7	Mavericks	Dallas	American Airlines Center
8	Nuggets	Denver	Pepsi Center
9	Pistons	Detroit	The Pallace at Auburn Hills
10	Warriors	Golden State	Oracle Arena
11	Rockets	Houston	Toyota Center
12	Pacers	Indiana	Bankers Life Fieldhouse
13	Clippers	Los Angeles	Staples Center

Second View

The Second view allows a user to see a player's points scored.

Statement

```
CREATE OR REPLACE VIEW PlayerPoint AS
SELECT
people.FirstName AS First_Name,
people.LastName AS Last_Name,
Team.TeamName AS Team_Name,
Team.TeamCity AS Team_City,
player_stats.Points AS player_points

FROM Team
INNER JOIN player_for ON team.teamID = player_for.teamID
INNER JOIN player ON player_for.pid = player.pid
INNER JOIN player_stats ON player.pid = player_stats.pid
INNER JOIN people ON Player_stats.pid = people.pid
GROUP BY people.FirstName, People.LastName, team.teamname, team.teamcity, player_stats.Points
ORDER BY people.LastName;
```

Output

```
SELECT * FROM PlayerPoint
```

:put pane

)ata Output | Explain | Messages | History

	first_name text	last_name text	team_name text	team_city text	player_points integer
1	Stephen	Curry	Warriors	Golden State	42
2	Isaiah	Thomas	Celtics	Boston	29

Stored Procedures

First Procedure

The First stored procedure allows a user to select a player by position.

Statement

```
CREATE OR REPLACE FUNCTION PlayersPosition (positions1 TEXT)
RETURNS TABLE ("First Name" TEXT, "Last Name" TEXT, "Position" TEXT)
AS
$$
BEGIN
RETURN Query
SELECT people.FirstName AS "First Name", people.LastName AS "LastName", positions.PositionName AS "Position"
FROM people
INNER JOIN player
ON people.PID = player.PID
INNER JOIN player_position
ON player.PID = player_position.PID
INNER JOIN positions
ON player_position.PositionID = positions.PositionID
WHERE positions.positionName = positions1
GROUP BY people.FirstName, people.LastName, positions.positionName
ORDER BY people.LastName;
END;
$$
LANGUAGE plpgsql;
```

Output

```
SELECT PlayersPosition ('Point Guard');
```

utput pane

Data Output | Explain | Messages | History

	playersposition record
1	(Stephen,Curry,"Point Guard")
2	(Chris,Paul,"Point Guard")
3	(Isaiah,Thomas,"Point Guard")

Second View

The Second View allows a user to select a team and see the current roster for that team.

Statement

```
CREATE OR REPLACE FUNCTION TeamRoster (Roster TEXT)
RETURNS TABLE("Team Name" TEXT, "First Name" TEXT, "Last Name" TEXT)
AS
$$
BEGIN
RETURN Query
SELECT team.teamName AS  "Team Name", people.FirstName AS "First Name", people.LastName AS "Last Name"
FROM people
INNER JOIN player_for ON people.pid = player_for.pid
INNER JOIN team ON player_for.teamID = team.teamID
WHERE Roster = team.teamname
GROUP BY people.firstname, people.lastname, team.teamname
ORDER BY people.lastname ASC;
END;
$$
LANGUAGE plpgsql;
```

Output

```
SELECT TeamRoster ('Celtics');
```

tput pane

Data Output | Explain | Messages | H

	teamroster record
1	(Celtics,Avery,Bradley)
2	(Celtics,Jae,Crowder)
3	(Celtics,Jared,Sullinger)
4	(Celtics,Isaiah,Thomas)
5	(Celtics,Evan,Turner)

Trigger

This trigger checks the old jersey number when updating a new jersey number for a player.

Statement

```
CREATE TRIGGER Jersey_Number
    BEFORE UPDATE ON Player_for
    FOR EACH ROW
    WHEN (OLD.JerseyNumberIS DISTINCT FROM NEW.JerseyNumber)
    EXECUTE PROCEDURE check_Jersey_Number();
```

Security

Database Admin Role

This role allows the database administrator to do anything he or she wants, including updating, deleting or inserting data into tables and revoking privileges.

Statement

```
CREATE ROLE databaseAdmin;
GRANT ALL PRIVILEGES
ON ALL TABLES IN SCHEMA PUBLIC
TO databaseAdmin;
```

General Admin Role

This role allows a general administrator to do things such as update rosters and update the games table.

Statement

```
CREATE ROLE generalAdmin;
GRANT INSERT, UPDATE, SELECT
ON ALL TABLES IN SCHEMA public
TO generalAdmin;
```

Public User Role

This role allows a public user to select the information they are looking for from the database.

Statement

```
CREATE ROLE publicUser;
GRANT SELECT
ON ALL TABLES IN SCHEMA public
TO publicUser;
```

Implementation Notes

- This database is strictly used for the National Basketball Association.
- Some queries will be needed to find the information. Not all information is readily available on hand.
- The draft table only shows the first selection of that draft year. However, you can find the selection of any player and the year he was drafted by joining other tables.

Known Problems

- One of the main problems of this database is that an admin will have to update the end dates for players and coaches if he is on the team in the beginning of each year.
- Another main problem is that you cannot calculate the total amount of points a team has scored throughout the season.

Future Enhancements

- I would like to include a playoffs table to be able to see playoffs from previous years.
- I would also like the change the team stats table to have the scores in two separate columns. These columns would now be integers and you could total points and look at efficiencies.
- I would also like to make the draft table include all selections so it is easier to see drafts by year.