THE NBA BUBBLE

By: Michael Wise
Database Design Project
CMPT 308
Dr. Alan Labouseur

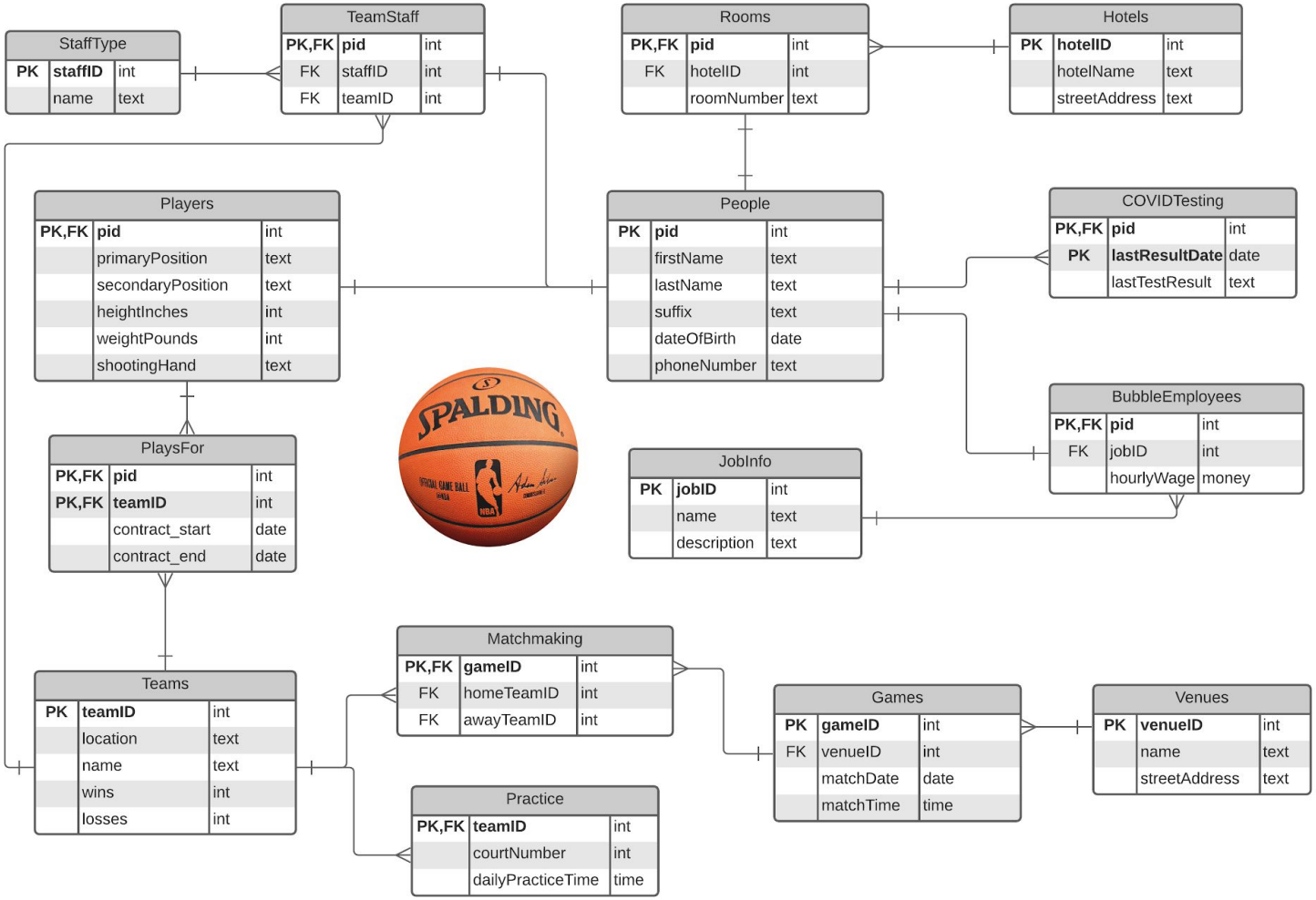
Table Of Contents

Executive Summary

This is a database for the NBA Bubble. In response to the pandemic, the National Basketball Association decided to finish the remainder of the 2019-2020 season in an isolation zone at Walt Disney World in attempt to protect its players. In order to maintain operation, the Disney campus would've needed a database to keep track of records relating to Players, Teams, COVID, Matches, Staff, etc.

This report will examine some key aspects of this database. Designed in PostgreSQL, users will be able to efficiently keep track of records for the remainder of the season. Included within this paper is an E/R diagram, create statements, inserted data, reports, views, stored procedures, triggers, and security measures. The end briefly reflects on some problems and future enhancements.

While the bubble has come to an end currently, the NBA still has plans to bring back the bubble format for the upcoming 2021 Playoffs, assuming the virus is still around.



THE NBA BUBBLE

ORLANDO, FL

DATABASE E/R DIAGRAM

MICHAEL WISE
CMPT 308

TABLES

People Table

Functional Dependencies:
pid → firstName, lastName, suffix, dateOfBirth, phoneNumber

Alan Labouseur is an upcoming NBA superstar. He is 8'2, making him the tallest player to ever play the game. Alan is so dominant on the court that people already consider him the GOAT.

In the bubble he averaged 46.5 points a night.

This table holds all of the people that interact in any way on the bubble campus. (players, staff, etc.)

```
CREATE TABLE People (
    pid             int  not null,
    firstName       text not null,
    lastName        text not null,
    suffix          text,
    dateOfBirth     date not null,
    phoneNumber     text,
 primary key(pid)
);
```

	pid [PK] integer	firstname text	lastname text	suffix text	dateofbirth date	phonenumber text
1	1	Lebron	James	[null]	1984-12-30	555-023-0824
2	2	Giannis	Antetokounm...	[null]	1994-12-06	555-123-4567
3	3	Kawhi	Leonard	[null]	1991-06-29	[null]
4	4	Anthony	Davis	[null]	1993-03-11	800-111-2323
5	5	John	Smith	Sr.	1964-11-06	321-100-2020
6	6	James	Harden	Jr.	1989-08-26	[null]
7	7	Damian	Lillard	[null]	1990-07-15	123-000-0000
8	8	Joel	Embiid	[null]	1994-03-16	[null]
9	9	Kemba	Walker	[null]	1990-05-08	555-150-0815
10	10	Frank	Vogel	[null]	1973-06-21	800-987-6543
11	11	Jimmy	Butler	III	1989-09-14	555-222-2222
12	12	Nick	Nurse	[null]	1967-07-24	123-426-3333
13	13	Thomas	Thomas	IX	1958-01-11	666-666-1337
14	14	Alan	Labouseur	[null]	1970-01-01	845-440-1102

Players Table

A table that keeps track of any people who are players, with their position(s), height, weight, and preferred shooting hand.

Functional Dependencies:
pid → primaryPosition, secondaryPostition, heightInches, weightPounds, shootingHand

Note that there is a check constraint on shootingHand, only allowing you to select left-handed, right-handed, or both.

```
CREATE TABLE Players (
    pid               int  not null references People(pid),
    primaryPosition   text not null,
    secondaryPosition text,
    heightInches      int,
    weightPounds      int,
    shootingHand      text not null check (shootingHand in ('left', 'right', 'both')),
 primary key(pid)
);
```

	pid [PK] integer	primaryposition text	secondaryposition text	heightinches integer	weightpounds integer	shootinghand text
1	1	SF	PG	81	250	both
2	2	PF	C	83	242	right
3	3	SF	[null]	79	225	right
4	4	C	PF	82	253	right
5	6	SG	PG	77	220	left
6	7	PG	[null]	74	195	right
7	8	C	[null]	84	280	right
8	9	PG	[null]	72	184	right
9	11	SF	SG	79	230	right
10	14	SG	C	98	360	both

COVIDTesting Table

Keeps a record of each person's latest test result. Positive tests are required to quarantine for two weeks and cannot play in any games for that amount of time.

Functional Dependencies:
(pid, lastResultDate) → lastTestResult

Note that there is a check constraint on lastTestResult. Only the only accepted values are 'positive' and 'negative'.

```
CREATE TABLE COVIDTesting (
   pid            int  not null references People(pid),
   lastResultDate date not null,
   lastTestResult text not null check (lastTestResult = 'positive' or lastTestResult = 'negative'),
 primary key(pid, lastResultDate)
);
```

	pid [PK] integer	lastresultdate [PK] date	lasttestresult text
1	1	2020-10-07	negative
2	2	2020-09-12	negative
3	3	2020-09-17	negative
4	4	2020-10-07	negative
5	5	2020-10-05	positive
6	6	2020-09-25	negative
7	7	2020-09-03	negative
8	8	2020-09-05	negative
9	9	2020-09-28	negative
10	10	2020-10-07	negative
11	11	2020-10-06	negative
12	12	2020-09-15	negative
13	13	2020-10-01	positive
14	14	2020-10-04	negative

JobInfo & BubbleEmployees Tables

Functional Dependencies:

JobInfo

- jobID → name, description

BubbleEmployees

- pid → jobID, hourlyWage

These tables store records related to the Disney staff. (janitors, food workers, hotel cast members, etc.)

```
CREATE TABLE JobInfo (
   jobID       int  not null,
   name        text not null,
   description text,
 primary key(jobID)
);
```

	jobid [PK] integer	name text	description text
1	1	Front Desk	Serves players and staff at the front desk.
2	2	Janitor	Cleans the public areas of the resorts.
3	3	Hotel Manager	The manager of a hotel.
4	4	Maid	Cleans rooms of players, coaches, and other staff.
5	5	Food Service	Provides and prepares food for teams.

```
CREATE TABLE BubbleEmployees (
   pid          int not null references People(pid),
   jobID        int not null references JobInfo(jobID),
   hourlyWage   money,
 primary key(pid)
);
```

	pid [PK] integer	jobid integer	hourlywage money
1	5	2	$11.30
2	13	1	$17.50

Hotels Table

Stores the list of all the hotels that players & staff are staying at throughout Disney World.

Functional Dependencies:

hotelID → hotelName, streetAddress

Each team stayed at one of 3 hotels: the Grand Floridian, Coronado Springs, and the Yacht Club. Unfortunately, they were closed to other guests through the duration of the bubble.

```
CREATE TABLE Hotels (
    hotelID       int not null,
    hotelName     text,
    streetAddress text,
 primary key(hotelID)
);
```

	hotelid [PK] integer	hotelname text	streetaddress text
1	1	Grand Floridian	4401 Floridian Way
2	2	Gran Destino Tower at Coronado Springs	1000 Buena Vista Dr
3	3	Disneys Yacht Club Resort	1700 Epcot Resorts Blvd

Rooms Table

Table that shows each person's assignment to a room in a hotel.

Functional Dependencies:

pid → hotelID, roomNumber

Imagine finding out that the room you're staying in was the same one Lebron James stayed in. How awesome would that be?

```
CREATE TABLE Rooms (
   pid        int not null references People(pid),
   hotelID    int not null references Hotels(hotelID),
   roomNumber text,
 primary key(pid)
);
```

	pid [PK] integer	hotelid integer	roomnumber text
1	2	2	3016
2	3	2	4120
3	4	2	5012
4	6	1	4428
5	7	3	1238
6	8	2	4014
7	9	2	1238
8	10	2	5008
9	11	2	3042
10	12	2	5764
11	14	1	1622
12	1	2	5004

Teams Table

Stores team names, along with their wins and losses for the 2020 regular season.

Functional Dependencies:

teamID → location, name, wins, losses

Fun fact: actually sad fact: the New York Knicks were not invited to the bubble because their record was so bad. Only teams that had a possibility of contending for the playoffs were invited.

I wouldn't be surprised if the Knicks didn't make the playoffs between right now and the time I die.

```
CREATE TABLE Teams (
    teamID   int  not null,
    location text not null,
    name     text not null,
    wins     int,
    losses   int,
 primary key(teamID)
);
```

	teamid [PK] integer	location text	name text	wins integer	losses integer
1	1	Los Angeles	Lakers	52	19
2	2	Milwaukee	Bucks	56	17
3	3	Toronto	Raptors	53	19
4	4	Los Angeles	Clippers	49	23
5	5	Houston	Rockets	44	28
6	6	Philadelphia	76ers	43	30
7	7	Portland	Trail Blazers	35	39
8	8	Boston	Celtics	48	24
9	9	Miami	Heat	44	29

PlaysFor Table

Functional Dependencies:
(pid, teamID) → contract_start, contract_end

I threw in their contracts for no other reason than to waste an hour on basketballreference.com, mainly staring at the salaries. Alan (pid 14) has recently signed a 94 year extension with the Toronto Raptors until the year 2112.

This table shows what players play for what teams. Because players can get traded and play for more than one team, this is an associative entity that solves the otherwise many-to-many relationship.

```
CREATE TABLE PlaysFor (
   pid            int not null references Players(pid),
   teamID         int not null references Teams(teamID),
   contract_start date,
   contract_end   date,
 primary key(pid, teamID)
);
```

	pid [PK] integer	teamid [PK] integer	contract_start date	contract_end date
1	1	1	2018-07-09	2022-07-09
2	2	2	2016-09-20	2021-09-20
3	3	4	2018-07-09	2022-07-09
4	4	1	2019-07-06	2025-12-03
5	6	5	2017-07-08	2023-07-08
6	7	7	2019-07-06	2025-07-06
7	8	6	2017-10-10	2023-10-10
8	9	8	2019-07-06	2023-07-06
9	11	9	2019-07-06	2023-07-06
10	14	3	2018-07-09	2112-07-09

StaffType & TeamStaff Tables

Functional Dependencies:

StaffType

- staffID → name

TeamStaff

- pid → staffID, teamID

These tables store information for the coaching, management, and training staff of each team. TeamStaff assigns each member their role and team.

```
CREATE TABLE StaffType (
   staffID int  not null,
   name    text not null,
 primary key (staffID)
);
```

	staffid [PK] integer	name text
1	1	Head Coach
2	2	Assistant Coach
3	3	General Manager
4	4	Athletic Trainer

```
CREATE TABLE TeamStaff (
   pid     int not null references People(pid),
   staffID int not null references StaffType(staffID),
   teamID  int not null references Teams(teamID),
 primary key(pid)
);
```

	pid [PK] integer	staffid integer	teamid integer
1	10	1	1
2	12	1	3

Practice Table

Functional Dependencies:
teamID → courtNumber, dailyPracticeTime

There is enough time in between each practice where no two teams should be sharing the same court at the same time. This also gives time for sanitization.

The Practice Table keeps records of when each team has daily practice. This includes what time they have it, and what practice court number they meet at.

```
CREATE TABLE Practice (
   teamID            int not null references Teams(teamID),
   courtNumber       int,
   dailyPracticeTime time,
 primary key(teamID)
);
```

	teamid [PK] integer	courtnumber integer	dailypracticetime time without time zone
1	1	1	07:30:00
2	2	2	08:00:00
3	3	3	08:30:00
4	4	4	09:00:00
5	5	1	09:30:00
6	6	2	10:00:00
7	7	3	10:30:00
8	8	4	11:00:00
9	9	1	11:30:00

Venues Table

Functional Dependencies:
venueID → name, streetAddress

This table is helpful for transporting teams to the right stadiums. They are all technically on the same complex so the streetAddress column is probably obsolete.

The bubble had 3 main venues on the ESPN World Wide of Sports Complex. This table assigns ID's to each of them and is referenced in the **Games** table.

```
CREATE TABLE Venues (
    venueID       int  not null,
    name          text,
    streetAddress text,
 primary key(venueID)
);
```

	venueid [PK] integer	name text	streetaddress text
1	1	HP Field House	701 S Victory Way
2	2	Visa Athletic Center	702 S Victory Way
3	3	AdventHealth Arena	700 S Victory Way

Games Table

Table that stores the information for each match, including their venue, date, and time. The table in the next slide is an associative entity that defines what teams are playing.

Functional Dependencies:

gameID → venueID, matchDate, matchTime

Fun fact: the 2020 Bubble had teams play 8 regular season games to determine seedings for the playoffs. The Phoenix Suns went 8-0 and still didn't qualify. Don't worry, they'll get their revenge.

guahahahaha

```
CREATE TABLE Games (
   gameID     int not null,
   venueID    int not null references Venues(venueID),
   matchDate  date,
   matchTime  time,
 primary key(gameID)
);
```

	gameid [PK] integer	venueid integer	matchdate date	matchtime time without time zone
1	1	3	2020-08-21	17:00:00
2	2	2	2020-08-21	21:00:00
3	3	1	2020-08-23	13:00:00
4	4	1	2020-08-28	18:30:00
5	5	3	2020-08-28	22:30:00
6	6	1	2020-09-06	15:00:00
7	7	2	2020-09-12	13:00:00

Matchmaking Table

Entity that decides who plays each other, including who is home and away for each matchup.

Functional Dependencies:

gameID → homeTeamID, awayTeamID

While each team did technically have home games, since they were all on the same campus, they had no traditional home court advantage in a sense that there were no fans in person.

```
CREATE TABLE Matchmaking (
   gameID     int not null references Games(gameID),
   homeTeamID int not null references Teams(teamID),
   awayTeamID int not null references Teams(teamID),
 primary key(gameID)
);
```

	gameid [PK] integer	hometeamid integer	awayteamid integer
1	1	1	4
2	2	2	3
3	3	5	7
4	4	8	9
5	5	6	2
6	6	4	5
7	7	9	1

VIEWS, REPORTS, STORED PROCEDURES, TRIGGERS, & SECURITY

View: PlayerInfo

```
create view PlayerInfo
as
select p.*, pl.primaryPosition, pl.secondaryPosition, pl.heightInches, pl.weightPounds, pl.shootingHand,
            f.contract_start, f.contract_end, t.location as teamCity, t.name as teamName
from People p inner join Players pl on p.pid = pl.pid
              inner join PlaysFor f on pl.pid = f.pid
              inner join Teams t    on f.teamID = t.teamID;

select * from PlayerInfo;
```

This view is simply a convenient way to view every stat pertaining to a player without having to create a million subqueries or joins. This includes player attributes, their team, and their contract information.

View: PlayerInfo (Sample Output)

pid integer	firstname text	lastname text	suffix text	dateofbirth date	phonenumber text	primaryposition text	secondaryposition text	heightinches integer	weightpounds integer	shootinghand text	contract_start date	contract_end date	teamcity text	teamname text
1	Lebron	James	[null]	1984-12-30	555-023-0824	SF	PG	81	250	both	2018-07-09	2022-07-09	Los Angeles	Lakers
2	Giannis	Antetokounm...	[null]	1994-12-06	555-123-4567	PF	C	83	242	right	2016-09-20	2021-09-20	Milwaukee	Bucks
3	Kawhi	Leonard	[null]	1991-06-29	[null]	SF	[null]	79	225	right	2018-07-09	2022-07-09	Los Angeles	Clippers
4	Anthony	Davis	[null]	1993-03-11	800-111-2323	C	PF	82	253	right	2019-07-06	2025-12-03	Los Angeles	Lakers
6	James	Harden	Jr.	1989-08-26	[null]	SG	PG	77	220	left	2017-07-08	2023-07-08	Houston	Rockets
7	Damian	Lillard	[null]	1990-07-15	123-000-0000	PG	[null]	74	195	right	2019-07-06	2025-07-06	Portland	Trail Blazers
8	Joel	Embiid	[null]	1994-03-16	[null]	C	[null]	84	280	right	2017-10-10	2023-10-10	Philadelphia	76ers
9	Kemba	Walker	[null]	1990-05-08	555-150-0815	PG	[null]	72	184	right	2019-07-06	2023-07-06	Boston	Celtics
11	Jimmy	Butler	III	1989-09-14	555-222-2222	SF	SG	79	230	right	2019-07-06	2023-07-06	Miami	Heat
14	Alan	Labouseur	[null]	1970-01-01	845-440-1102	SG	C	98	360	both	2018-07-09	2112-07-09	Toronto	Raptors

This view is simply a convenient way to view every stat pertaining to a player without having to create a million subqueries or joins. This includes player attributes, their team, and their contract information.

View: PowerRankings

```
create view PowerRankings
as
select *, cast((wins * 1.0 /(wins + losses)) as decimal(4,3)) as winPercentage
from Teams
order by winPercentage DESC;

select * from PowerRankings;
```

This view calculates every team's Win Percentage. This is evaluated by taking the your total wins divided by the total games played. Next, the view appends the percentage to each team's standings and ranks them from best record to worst record.

View: PowerRankings (Sample Output)

	teamid integer	location text	name text	wins integer	losses integer	winpercentage numeric (4,3)
1	2	Milwaukee	Bucks	56	17	0.767
2	3	Toronto	Raptors	53	19	0.736
3	1	Los Angeles	Lakers	52	19	0.732
4	4	Los Angeles	Clippers	49	23	0.681
5	8	Boston	Celtics	48	24	0.667
6	5	Houston	Rockets	44	28	0.611
7	9	Miami	Heat	44	29	0.603
8	6	Philadelphia	76ers	43	30	0.589
9	7	Portland	Trail Blazers	35	39	0.473

This view calculates every team's Win Percentage. This is evaluated by taking the your total wins divided by the total games played. Next, the view appends the percentage to each team's standings and ranks them from best record to worst record.

Reports

Lists the total number of players whose contracts are expiring within the next two off-seasons (up to summer 2022).

```
select count(pid)
from PlaysFor
where contract_end < '2022-08-01';
```

	count bigint
1	3

Reports the total number of players/staff who have tested positive for COVID-19 and must quarantine.

```
select count(pid)
from COVIDTesting
where lastTestResult = 'positive';
```

	count bigint
1	2

Stored Procedure: showSchedule

```
create or replace function showSchedule(text, REFCURSOR) returns refcursor as
$$
declare
   teamName  text      := $1;
   resultset REFCURSOR := $2;
begin
   open resultset for
      select m.gameID, t.name as homeTeam, t2.name as awayTeam, g.matchDate, g.matchTime, v.name as Venue, v.streetAddress
      from Matchmaking m inner join Games g  on m.gameID   = g.gameID
                         inner join Venues v on g.venueID  = v.venueID
                         inner join Teams t  on t.teamID   = m.homeTeamID
                         inner join Teams t2 on t2.teamID  = m.awayTeamID
      where t.name = teamName or t2.name = teamName;
   return resultset;
end;
$$
language plpgsql;
```

This function will show all of the upcoming games for a specified team. It also outputs the match info (i.e. the venue, time, and your opposing team). This is good procedure for coaches to use.

Stored Procedure: showSchedule (Output)

```
select showSchedule('Clippers', 'results');
fetch all from results;
```

	gameid integer	hometeam text	awayteam text	matchdate date	matchtime time without time zone	venue text	streetaddress text
1	1	Lakers	Clippers	2020-08-21	17:00:00	AdventHealth Arena	700 S Victory Way
2	6	Clippers	Rockets	2020-09-06	15:00:00	HP Field House	701 S Victory Way

In this example, the Clippers have two games coming up - one vs. Lakers and the other vs. Rockets.

Stored Procedure: hotelSearch

```
create or replace function hotelSearch(text, REFCURSOR) returns refcursor as
$$
declare
   nameOfHotel  text      := $1;
   resultset    REFCURSOR := $2;
begin
   open resultset for
      select p.*, r.roomNumber, h.hotelName, h.streetAddress
      from People p inner join Rooms r on p.pid = r.pid
                    inner join Hotels h on r.hotelID = h.hotelID
      where h.hotelName = nameOfHotel;
   return resultset;
end;
$$
language plpgsql;
```

This procedure returns a list of all of the people staying in the specified hotel. It also gives their room/contact information. People at the front desk could use this if they need to contact players/staff.

Stored Procedure: hotelSearch (Sample output)

```
select hotelSearch('Grand Floridian', 'results');
fetch all from results;
```

	pid integer	firstname text	lastname text	suffix text	dateofbirth date	phonenumber text	roomnumber text	hotelname text	streetaddress text
1	6	James	Harden	Jr.	1989-08-26	[null]	4428	Grand Floridian	4401 Floridian Way
2	14	Alan	Labouseur	[null]	1970-01-01	845-440-1102	1622	Grand Floridian	4401 Floridian Way

We can obtain a list of every person staying in the Grand Floridian. In this example, James Harden and Alan Labouseur are staying there.

Trigger: isTooShort()

This trigger will delete anyone entered into the People table that is less than 5'3. This is the NBA. If you are under 5'3 you do not deserve to be in the NBA. You will get destroyed. Only Muggsy Bogues was able to pull that off. Sorry.

We tried adding 4'11 (59 inches, pid=5) John Smith to the table. He was instantly deleted.

```
create or replace function isTooShort() returns trigger as
$$
begin
   if (NEW.heightInches < 63) then
   delete from Players where heightInches = NEW.heightInches;
   end if;

   return new;
end;
$$
language plpgsql;

create trigger isTooShort
after insert on Players
for each row
execute procedure isTooShort();
```

```
INSERT INTO Players (pid, primaryPosition, secondaryPosition, heightInches, weightPounds, shootingHand)
VALUES
 (005, 'PG', NULL, 59, 65, 'left');
select * from Players;
```

	pid [PK] integer	primaryposition text	secondaryposition text	heightinches integer	weightpounds integer	shootinghand text
1	1	SF	PG	81	250	both
2	2	PF	C	83	242	right
3	3	SF	[null]	79	225	right
4	4	C	PF	82	253	right
5	6	SG	PG	77	220	left
6	7	PG	[null]	74	195	right
7	8	C	[null]	84	280	right
8	9	PG	[null]	72	184	right
9	11	SF	SG	79	230	right
10	14	SG	C	98	360	both

Trigger: roomFull()

Output:Thomas Thomas was unfortunately not added. He was excited to finally meet Lebron ;(

	pid [PK] integer	hotelid integer	roomnumber text
1	2	2	3016
2	3	2	4120
3	4	2	5012
4	6	1	4428
5	7	3	1238
6	8	2	4014
7	9	2	1238
8	10	2	5008
9	11	2	3042
10	12	2	5764
11	14	1	1622
12	1	2	5004

```
create or replace function roomFull() returns trigger as
$$
begin
   if (NEW.roomNumber in (select roomNumber from Rooms)) then
   delete from Rooms where roomNumber = NEW.roomNumber;
   end if;

   return new;
end;
$$
language plpgsql;
```

```
create trigger roomFull
after insert on Rooms
for each row
execute procedure roomFull();
```

```
-- Let's try to add Thomas Thomas to Lebron's room. --
INSERT INTO Rooms (pid, hotelID, roomNumber)
VALUES
 (013, 02, 5004);
select * from Rooms
```

This trigger prevents you from adding someone to a hotel room that is already occupied by somebody else.

Security (Roles)


```
create role admin;
grant all on
all tables
in schema public
to admin;
```

Admin: has access to everything in the database.

```
create role frontDeskWorker;
grant select, insert, update
on Rooms, Hotels
to frontDeskWorker;
```

Front Desk: has ability to update/change rooming information for guests.

```
create role coach;
grant select
on Players, Teams, TeamStaff, StaffType, Practice, Matchmaking,
to coach;
```

Coach: can view most player/staff/game info

```
create role healthDept;
grant select, insert, update
on COVIDTesting
to healthDept;
```

Health Worker: can update the database with the latest COVID info/results, but nothing else

Implementation Notes

- While it seems like the bubble was a “one-and-done” event, there still might be plans to bring back for the future. The NBA actually lost billions of dollars of revenue by losing the second half of last season.
- Of course, the data inserted is not complete. If I had the time to include all the rest of the teams, players, and staff, I could probably create some even more interesting queries than the ones I had.
- I liked the combination of the sports analytics with the hotel management side of things. This type of database could definitely work at a sports complex that also happens to have hotels.
- This project was way too much fun, but I did waste a lot of time looking up statistics online for players when it was probably better to just guess.

Known Problems & Future Enhancements

- This database currently only works for the 2020 season. I would need to add more tables to add functionality for future seasons or other leagues (i.e. the MLS was also here).
- I got a little confused with the ordering of the tables when doing create/insert statements, but it should be correct now.
- The roomFull() procedure is kind of buggy. I swear I had it working and now it deletes unintended rows.
- For the future, I could've renamed the attributes to be more consistent. To me, "pid" just sounds kind of weak for a system like this.
- I want to add player performance statistics like points, assists, rebounds, steals, and blocks. I chose not to for this specific snapshot because it would've been too time consuming.
- Alan Labouseur is a legend.