Hogwarts Inter-House Quidditch Cup DB

-- This database was brought to you by: G Leaden

Table of Contents:

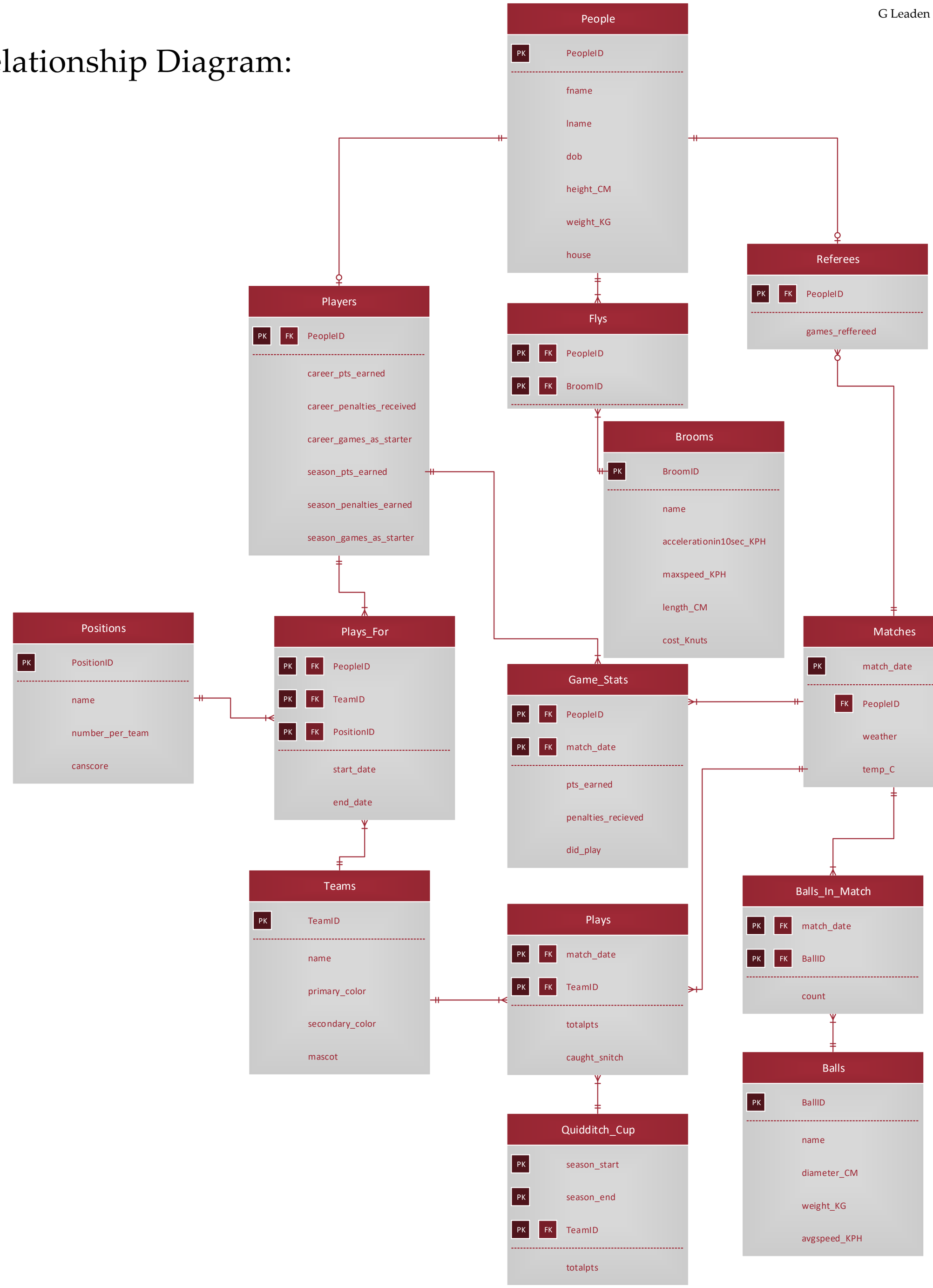
Executive Summary

Hogwarts contracted me to make this database after an awful magical fire that blazed through Madam Hooch's offices, destroying nearly all of the existing data of the Hogwarts Inter-House Quidditch Cup. A longstanding tradition, the Hogwarts Inter-House Quidditch Cup needed a database, with multiple backups, to keep records safe and accurate.

This paper outlines such a database. Designed in postgres, this database aims to simplify data entry for the user while maintaining a significant level of complexity and offering various queries, views, and stored procedures with which to update/modify the database. Included within this paper is an ER diagram, various tables and queries on those tables with sample data in place, views, stored procedures, triggers, roles, notes on the implementation of the database, some insight into my thought process, and lastly, known problems within the database and some possible future enhancement suggestions.

The Inter-House Quidditch Cup is played with four teams, each representing one of the four houses from Hogwarts. Each team consists of seven players flying around on brooms. There are balls to catch, dodge, and score. Each match is played until the snitch is caught, granting 150 pts to the catching team.

Entity-Relationship Diagram:



Types:

house Type - The house type contains text entries of the titles of the four houses

```
CREATE TYPE house AS ENUM ('Gryffindor', 'Hufflepuff', 'Ravenclaw', 'Slytherin');
```

wcondition Type - The wcondition type contains text entries of various appropriate weather conditions

```
CREATE TYPE wcondition AS ENUM ('clear','rainy','thunderstormy','snowy','windy','cloudy');
```

Tables:

people Table - The people table keeps track of any person within the database with basic information. The attributes in this table are shared with both players and referees.

```
CREATE TABLE people(
    PeopleID  int           not null,
    fname     text          not null,
    lname     text          not null,
    dob       DATE          not null,
    height_cm decimal(8,3) not null,
    weight_kg decimal(7,3) not null,
    house     house         not null,
    primary key(PeopleID)
);
```

Functional Dependencies:

peopleid -> fname, lname, dob, height_cm, weight_kg, house

	peopleid integer	fname text	lname text	dob date	height_cm numeric(8,3)	weight_kg numeric(7,3)	house house
1	1	marcus	zimmermann	1997-01-22	167.640	63.503	Hufflepuff
2	2	alan	labouseur	1970-01-01	9999.000	9999.000	Slytherin
3	3	rolanda	hooch	1900-01-01	160.020	50.802	Hufflepuff
4	4	skittles	taylor	1997-03-07	165.100	65.317	Slytherin
5	5	jeff	lupia	1997-09-14	199.581	102.058	Gryffindor
6	6	anton	zimmermann	1997-01-21	167.650	63.504	Slytherin
7	7	ron	weasley	1980-02-29	172.720	66.830	Gryffindor
8	8	fred	weasley	1978-03-31	190.500	87.870	Gryffindor
9	9	george	weasley	1978-03-31	190.500	87.870	Gryffindor
10	10	ginerva	weasley	1981-08-10	167.640	61.440	Gryffindor
11	11	charlie	weasley	1972-12-11	182.880	88.462	Gryffindor
12	12	oliver	wood	1975-10-04	175.260	83.124	Gryffindor
13	13	james	potter	1960-03-26	177.800	77.651	Gryffindor
14	14	dean	thomas	1980-06-10	190.500	80.000	Gryffindor
15	15	john	doe	1999-11-19	188.960	87.332	Slytherin
16	16	john	deer	1998-03-04	166.388	59.140	Slytherin
17	17	mary	smith	1998-06-05	120.700	72.991	Slytherin
18	18	doug	smith	1997-02-04	199.691	101.803	Slytherin
19	19	sample	data	1999-12-31	177.800	80.556	Slytherin
20	20	albus	weasley	1999-02-28	148.371	81.584	Gryffindor
21	21	the	doctor	1066-12-24	183.439	95.467	Hufflepuff

Table I: people

players Table - The players table keeps track of any person within the database who is also a player, the table keeps stats of that player both for the season and for their career.

```
CREATE TABLE players(
    PeopleID                        int not null references people(PeopleID),
    career_pts_earned               int,
    career_penalties_recieved       int,
    career_games_played_as_starter int,
    season_pts_earned               int,
    season_penalties_recieved       int,
    season_games_played_as_starter int,
    primary key(PeopleID)
);
```

Functional Dependencies:

peopleid -> career_pts_earned, career_penalties_recieved, career_games_played_as_starter, season_pts_earned, season_penalties_recieved, season_games_played_as_starter

	peopleid integer	career_pts_earned integer	career_penalties_recieved integer	career_games_played_as_starter integer	season_pts_earned integer	season_penalties_recieved integer	season_games_played_as_starter integer
1	33						
2	29	830	0	3	830	0	3
3	1	0	1	3	0	1	3
4	30	0	3	3	0	3	3
5	31	1280	2	3	1280	2	3
6	21	0	0	3	0	0	3
7	25	300	0	3	300	0	3
8	26	0	1	3	0	1	3
9	15	0	2	3	0	2	3
10	4	300	0	3	300	0	3
11	6	10	3	3	10	3	3
12	19	40	4	3	40	4	3
13	18	100	0	3	100	0	3
14	17	0	1	3	0	1	3
15	16	0	2	3	0	2	3
16	37	400	2	3	400	2	3
17	20	450	3	3	450	3	3
18	35	0	4	3	0	4	3
19	34	0	0	3	0	0	3
20	27	520	0	3	520	0	3
21	32	110	0	3	110	0	3

Table II: players

referees Table - The referees table keeps track of any person within the database who is also a referee, the table also counts how many games each referee has participated in.

```
CREATE TABLE referees(
    PeopleID        int   not null references people(PeopleID),
    games_refereed  int   not null,
    primary key(PeopleID)
);
```

Functional Dependencies:

peopleid -> games_refereed

	peopleid integer	games_refereed integer
1	2	42
2	3	265

Table III: referees

positions Table - The positions table keeps track of all positions a player can be in the game Quidditch, giving each a name, the amount of players on a team with that position, and whether or not a player in that position can score.

```
CREATE TABLE positions(
      PositionID      int     not null,
      name            text    not null,
      number_per_team int     not null,
      canscore        boolean not null,
      primary key(PositionID)
);
```

Functional Dependencies:

positionid -> name, number_per_team, canscore

	positionid integer	name text	number_per_team integer	canscore boolean
1	1	beater	2	f
2	2	chaser	3	t
3	3	keeper	1	f
4	4	seeker	1	t

Table IV: positions

brooms Table - The brooms table keeps track of all the brooms a person can fly during the game. This table gives each broom a name and then keeps stats such as acceleration, max speed, length of broom, and the broom's cost.

```
CREATE TABLE brooms(
      BroomID                 int          not null,
      name                    text         not null,
      accelerationin10sec_KPH decimal(7,3) not null,
      maxspeed_KPH            decimal(7,3) not null,
      length_CM               decimal(7,3) not null,
      cost_Knuts              int          not null,
      primary key(BroomID)
);
```

Functional Dependencies:

broomid -> name, accelerationin10sec_KPH, maxspeed_KPH, length_CM, cost_Knuts

	broomid integer	name text	accelerationin10sec_kph numeric(7,3)	maxspeed_kph numeric(7,3)	length_cm numeric(7,3)	cost_knuts integer
1	1	Nimbus 2000	144.841	241.402	155.880	167620
2	2	Nimbus 2001	177.028	273.588	150.667	197200
3	3	Firebolt	241.402	354.056	153.718	4930000
4	4	Comet 180	65.886	110.692	201.222	46835
5	5	Comet 290	96.561	180.247	160.005	128180
6	6	Cleansweep 7	90.603	168.995	175.374	118320
7	7	Cleansweep 11	112.654	197.949	158.713	147900

Table V: brooms

flys Table - The flys table keeps track of what broom each person flys and relates them together creating a many to many relationship between people and brooms.

```
CREATE TABLE flys(
    PeopleID int     not null references people(PeopleID),
    BroomID  int     not null references brooms(BroomID),
    primary key(PeopleID, BroomID)
);
```

Functional Dependencies:

broomid -> peopleid

	peopleid integer	broomid integer
1	1	1
2	2	1
3	3	4
4	4	1
5	5	1
6	6	3
7	7	4
8	8	6
9	9	6
10	10	4
11	11	6
12	12	6
13	13	4
14	14	7
15	15	2
16	16	2
17	17	2
18	18	2
19	19	2
20	20	4
21	21	1
22	22	2
23	23	1
24	24	4
25	25	1
26	26	4
27	27	2
28	28	7
29	29	2
30	30	3

Table VI: flys

teams Table - The teams table keeps track of every team in the league, along with stats for that team such as team colors, team mascot, and the team name.

```
CREATE TABLE teams(
    TeamID          int     not null,
    name            house   not null,
    primary_color   text    not null,
    secondary_color text    not null,
    mascot          text    not null,
    primary key(TeamID)
);
```

Functional Dependencies:

teamid -> name, primary_color, secondary_color, mascot

	teamid integer	name house	primary_color text	secondary_color text	mascot text
1	1	Gryffindor	crimson	gold	lion
2	2	Hufflepuff	mustard	black	badger
3	3	Ravenclaw	royal blue	bronze	eagle
4	4	Slytherin	green	silver	serpent

Table VII: teams

plays_for Table - The plays_for table intersects players, teams, and positions in order to keep track of every player in the league, what team they are on, and what position they have on the team throughout their career.

```
CREATE TABLE plays_for(
    PeopleID   int     not null references people(PeopleID),
    TeamID     int     not null references teams(TeamID),
    PositionID int     not null references positions(PositionID),
    start_date DATE    not null,
    end_date   DATE,
    primary key(PeopleID, TeamID, PositionID)

);
```

Functional Dependencies:

peopleid, teamid, positionid -> start_date, end_date

	peopleid integer	teamid integer	positionid integer	start_date date	end_date date
1	1	2	1	2015-08-31	
2	4	4	4	2015-08-31	
3	5	1	1	2015-08-31	
4	6	4	2	2015-08-31	
5	7	1	3	1995-08-31	1997-06-01
6	8	1	1	1990-08-31	1995-06-01
7	9	1	1	1990-08-31	1995-06-01
8	10	1	2	1996-08-31	1997-06-01
9	10	1	4	1995-08-31	1996-06-01
10	11	1	4	1985-08-31	1991-06-01
11	12	1	3	1988-08-31	1993-06-01
12	13	1	2	1974-08-31	1975-06-01
13	14	1	2	1996-08-31	1997-06-01
14	15	4	1	2014-08-31	
15	16	4	1	2015-08-31	
16	17	4	3	2015-08-31	
17	18	4	2	2013-08-31	
18	19	4	2	2013-08-31	
19	20	1	4	2015-08-31	
20	21	2	3	2015-08-31	
21	22	4	4	1992-08-31	1997-06-01
22	23	2	4	1993-08-31	1995-06-01
23	24	4	1	1972-08-31	1980-06-01
24	25	2	2	2015-08-31	
25	26	2	4	2015-08-31	
26	27	1	2	2014-08-31	
27	28	3	3	2015-08-31	
28	29	2	2	2014-08-31	
29	30	2	1	2015-08-31	
30	31	2	2	2015-08-31	

Table VIII: plays_for

balls Table - The balls table contains all data for the balls used in the game, including possible speeds (bewitched balls fly on their own), weight, and diameter.

```
CREATE TABLE balls(
      BallID       int          not null,
      name         text         not null,
      diameter_CM  decimal(7,3) not null,
      weight_KG    decimal(7,3) not null,
      avgspeed_KPH decimal(7,3),
      primary key(BallID)
);
```

Functional Dependencies:

ballid -> name, diameter_CM, weight_KG, avgspeed_KPH

	ballid integer	name text	diameter_cm numeric(7,3)	weight_kg numeric(7,3)	avgspeed_kph numeric(7,3)
1	1	Quaffle	30.480	2.268	
2	2	Bludger	25.400	66.885	97.204
3	3	Snitch	4.445	3.040	360.000

Table IX: balls

matches Table - The matches table contains data for each match played at Hogwarts, including the date (only one match can be held per day), people in the match, and weather conditions.

```
CREATE TABLE matches(
      match_date DATE         not null,
      peopleid   int          not null references people(peopleid),
      weather    wcondition   not null,
      temp_C     decimal(7,3) not null,
      primary key(match_date)
);
```

Functional Dependencies:

match_date -> peopleid, weather, temp_C

	match_date date	peopleid integer	weather wcondition	temp_c numeric(7,3)
1	2015-11-07	3	clear	7.222
2	2015-11-28	3	cloudy	2.778
3	2015-12-25	3	snowy	0.000
4	2016-02-20	3	clear	3.333
5	2016-03-12	3	rainy	8.889
6	2016-05-07	3	clear	8.889
7	2016-05-28	3	clear	12.778
8	1996-05-25	3	clear	25.000
9	1995-02-04	3	snowy	-1.512
10	1988-11-05	3	rainy	2.210
11	1975-05-25	3	cloudy	9.874
12	1997-02-24	3	thunderstormy	4.665
13	1992-11-05	3	thunderstormy	9.644
14	1994-03-15	3	clear	2.843

Table X: matches

balls_in_match Table - The balls_in_match table contains data on how many of each ball were present during a given match. This table supports the many to many relationship between balls and matches.

```
CREATE TABLE balls_in_match(
      match_date DATE not null references matches(match_date),
      BallID     int  not null references balls(BallID),
      count      int  not null,
      primary key(match_date, BallID)
);
```

Functional Dependencies:

match_date, ballid -> count

	match_date date	ballid integer	count integer
1	2016-05-28	1	1
2	2016-05-28	2	2
3	2016-05-28	3	1
4	2016-05-07	1	1
5	2016-05-07	2	2
6	2016-05-07	3	1
7	2016-03-12	1	1
8	2016-03-12	2	2
9	2016-03-12	3	1
10	2016-02-20	1	1
11	2016-02-20	2	2
12	2016-02-20	3	1
13	2015-12-25	1	1
14	2015-12-25	2	2
15	2015-12-25	3	1
16	2015-11-28	1	1
17	2015-11-28	2	2
18	2015-11-28	3	1
19	2015-11-07	1	1
20	2015-11-07	2	2
21	2015-11-07	3	1
22	1997-02-24	1	1
23	1997-02-24	2	2
24	1997-02-24	3	1
25	1996-05-25	1	1
26	1996-05-25	2	2
27	1996-05-25	3	1

Table XI: balls_in_match

plays Table - The plays table contains data about which teams played in which match, along with their respective points earned for the match and whether or not they caught the snitch.

```
CREATE TABLE plays(
      match_date       DATE    not null references matches(match_date),
      TeamID           int     not null references teams(TeamID),
      totalpts         int     not null,
      caught_snitch    boolean not null,
      primary key(match_date, TeamID)
);
```

Functional Dependencies:

match_date, teamid -> totalpts, caught_snitch

	match_date date	teamid integer	totalpts integer	caught_snitch boolean
1	2015-11-07	1	360	t
2	2015-11-07	4	300	t
3	2015-11-28	2	1390	f
4	2015-11-28	3	1540	t
5	2016-02-20	3	1520	t
6	2016-02-20	4	150	f
7	2016-03-12	1	940	t
8	2016-03-12	2	1540	f
9	2016-05-07	2	310	f
10	2016-05-07	4	150	t
11	2016-05-28	1	480	t
12	2016-05-28	3	920	f
13	1995-02-04	1	0	f
14	1996-05-25	1	150	f
15	1988-11-05	1	300	f
16	1975-05-25	1	420	f
17	1997-02-24	1	360	f
18	1992-11-05	4	0	f
19	1994-03-15	2	300	f
20	1975-05-25	4	0	f

Table XII: plays

game_stats Table - The game_stats table contains stats about each player playing in a match, including points earned, penalties received, and whether or not that player did in fact play for their team that match.

```
CREATE TABLE game_stats(
     PeopleID           int not null references
people(PeopleID),
     match_date         DATE not null references
matches(match_date),
     pts_earned         int not null,
     penalties_recieved int not null,
     did_play           boolean not null,
     primary key(PeopleID, match_date)
);
```

Functional Dependencies:

peopleid, match_date -> pts_earned, penalties_recieved, did_play

- Fun Fact!
  Whenever data is inserted into this table, it auto populates THREE separate tables with data from this!
  (see stored procedures and triggers to learn more)

	peopleid integer	match_date date	pts_earned integer	penalties_recieved integer	did_play boolean
1	5	2015-11-07	0	6	t
2	20	2015-11-07	150	2	t
3	27	2015-11-07	100	0	t
4	32	2015-11-07	10	0	t
5	35	2015-11-07	0	1	t
6	34	2015-11-07	0	0	t
7	37	2015-11-07	100	1	t
8	4	2015-11-07	150	0	t
9	6	2015-11-07	0	3	t
10	19	2015-11-07	0	4	t
11	18	2015-11-07	0	0	t
12	17	2015-11-07	0	1	t
13	16	2015-11-07	0	1	t
14	15	2015-11-07	0	1	t
15	29	2015-11-28	300	0	t
16	1	2015-11-28	0	0	t
17	30	2015-11-28	0	1	t
18	26	2015-11-28	0	0	t
19	31	2015-11-28	580	0	t
20	21	2015-11-28	0	0	t
21	25	2015-11-28	210	0	t
22	41	2015-11-28	410	0	t
23	42	2015-11-28	480	0	t
24	28	2015-11-28	0	0	t
25	36	2015-11-28	150	0	t
26	38	2015-11-28	0	0	t
27	39	2015-11-28	90	0	t

Table XIII: game_stats

quidditch_cup Table - The quidditch_cup table contains information about every season (starting in November and ending in May) of the Hogwarts Inter-House Quidditch Cup. If there is not a full season recorded, totalpts will return 0

```
CREATE TABLE quidditch_cup(
    season_start int not null,
    season_end   int not null,
    TeamID       int not null references teams(TeamID),
    totalpts     int not null,
    primary key(season_start, season_end, TeamID)
);
```

Functional Dependencies:

season_start, season_end, teamid -> totalpts

	season_start integer	season_end integer	teamid integer	totalpts integer
1	1995	1996	1	0
2	1994	1995	1	0
3	1988	1989	1	0
4	1974	1975	1	0
5	1996	1997	1	0
6	1992	1993	4	0
7	1993	1994	2	0
8	1974	1975	4	0
9	2015	2016	1	1780
10	2015	2016	2	3240
11	2015	2016	3	3980
12	2015	2016	4	600

Table XIV: quidditch_cup

Views:

current_rosters - Lists the name, position, team, and broom for every active player on every team.

```
CREATE VIEW current_rosters as
      SELECT   fname as first_name,lname as last_name,positions.name as position, teams.name as team,
brooms.name as current_broom
      FROM     players
             INNER JOIN people    ON players.peopleid=people.peopleid
             INNER JOIN plays_for ON players.peopleid=plays_for.peopleid
             INNER JOIN positions ON plays_for.positionid=positions.positionid
             INNER JOIN teams     ON plays_for.teamid=teams.teamid
             INNER JOIN flys      ON players.peopleid=flys.peopleid
             INNER JOIN brooms    ON flys.broomid=brooms.broomid
      WHERE    plays_for.end_date   IS NULL
      ORDER BY teams ASC, people ASC;
```

	first_name text	last_name text	position text	team house	current_broom text
1	jeff	lupia	beater	Gryffindor	Nimbus 2000
2	albus	weasley	seeker	Gryffindor	Comet 180
3	son	goku	chaser	Gryffindor	Nimbus 2001
4	kevin	kliendshmidt	chaser	Gryffindor	Cleansweep 11
5	reingald	weasley	beater	Gryffindor	Nimbus 2001
6	regina	weasley	beater	Gryffindor	Nimbus 2001
7	soren	bjerg	chaser	Gryffindor	Firebolt
8	marcus	zimmermann	beater	Hufflepuff	Nimbus 2000
9	the	doctor	keeper	Hufflepuff	Nimbus 2000
10	ian	sniffen	chaser	Hufflepuff	Nimbus 2000
11	dank	memes	seeker	Hufflepuff	Comet 180
12	son	goten	chaser	Hufflepuff	Nimbus 2001
13	troy	capybara	beater	Hufflepuff	Firebolt
14	myrtle	scamander	chaser	Hufflepuff	Cleansweep 11
15	son	gohan	keeper	Ravenclaw	Cleansweep 11
16	gabe	newell	seeker	Ravenclaw	Firebolt
17	vincent	wang	beater	Ravenclaw	Nimbus 2000
18	william	li	chaser	Ravenclaw	Comet 290
19	joshua	leesman	beater	Ravenclaw	Comet 290
20	marcus-anton	zimmermann	chaser	Ravenclaw	Cleansweep 11
21	the	vision	chaser	Ravenclaw	Firebolt
22	skittles	taylor	seeker	Slytherin	Nimbus 2000
23	anton	zimmermann	chaser	Slytherin	Firebolt
24	john	doe	beater	Slytherin	Nimbus 2001
25	john	deer	beater	Slytherin	Nimbus 2001
26	mary	smith	keeper	Slytherin	Nimbus 2001
27	doug	smith	chaser	Slytherin	Nimbus 2001
28	sample	data	chaser	Slytherin	Nimbus 2001

View I: current_rosters

highest_scorer - Lists the player with the highest career points earned, along with their team and their current broom

```
CREATE VIEW highest_scorer as
      SELECT   fname as first_name,lname as last_name, career_pts_earned as total_points,
               positions.name as position, teams.name as team, brooms.name as current_broom
      FROM     players
                 INNER JOIN people    ON players.peopleid=people.peopleid
                 INNER JOIN plays_for ON players.peopleid=plays_for.peopleid
                 INNER JOIN positions ON plays_for.positionid=positions.positionid
                 INNER JOIN teams     ON plays_for.teamid=teams.teamid
                 INNER JOIN flys      ON players.peopleid=flys.peopleid
                 INNER JOIN brooms    ON flys.broomid=brooms.broomid
      WHERE    players.career_pts_earned = (SELECT   players.career_pts_earned
                                            FROM     players
                                            WHERE    career_pts_earned IS NOT NULL
                                            ORDER BY career_pts_earned DESC
                                            LIMIT 1)
      ORDER BY career_pts_earned DESC, teams ASC;
```

	first_name text	last_name text	total_points integer	position text	team house	current_broom text
1	myrtle	scamander	1280	chaser	Hufflepuff	Cleansweep 11

View II: highest_scorer

Reports:

1. All people who are shorter than their brooms

```
SELECT fname as first_name, lname as last_name, brooms.name as broom
FROM   people
  INNER JOIN flys on people.peopleid=flys.peopleid
  INNER JOIN brooms on flys.broomid=brooms.broomid
WHERE people.height_CM < brooms.length_CM;
```

	first_name text	last_name text	broom text
1	rolanda	hooch	Comet 180
2	ron	weasley	Comet 180
3	ginerva	weasley	Comet 180
4	oliver	wood	Cleansweep 7
5	james	potter	Comet 180
6	mary	smith	Nimbus 2001
7	albus	weasley	Comet 180
8	cedric	diggory	Nimbus 2000
9	bartemius	crouch	Comet 180
10	dank	memes	Comet 180
11	kevin	kliendshmidt	Cleansweep 11
12	gabe	newell	Firebolt
13	soren	bjerg	Firebolt
14	william	li	Comet 290
15	joshua	leesman	Comet 290

Report 1

2. Query to return brooms whose max speed is slower than a snitch

```
SELECT name as broom, brooms.maxspeed_KPH
FROM   brooms
WHERE  brooms.maxspeed_KPH < (SELECT avgspeed_KPH
                              FROM   balls
                              WHERE   balls.ballid=3);
```

	broom text	maxspeed_kph numeric(7,3)
1	Nimbus 2000	241.402
2	Nimbus 2001	273.588
3	Comet 180	110.692
4	Comet 290	180.247
5	Cleansweep 7	168.995
6	Cleansweep 11	197.949

Report 2

BONUS. Query to put the basic stats of peopleid(26) and peopleid(2) side by side

```
select * from people where peopleid=26 OR peopleid=2;
```

	peopleid integer	fname text	lname text	dob date	height_cm numeric(8,3)	weight_kg numeric(7,3)	house house
1	2	alan	labouseur	1970-01-01	9999.000	9999.000	Slytherin
2	26	dank	memes	2012-01-31	1.000	1.000	Hufflepuff

Report 3

Stored Procedures:

gamedata_to_plays - This function returns a trigger and takes all of the data from the game_stats table and uses it to populate the plays table.

```
CREATE OR REPLACE FUNCTION gamedata_to_plays() RETURNS trigger AS $to_plays$
    DECLARE
            my_teamID          int;
            totalpoints        int;
            snitch             boolean;
            seekerid       int;
    BEGIN
            snitch=false;
            my_teamID=(SELECT teamID
                    FROM   plays_for
                      WHERE  NEW.PeopleID=plays_for.PeopleID AND ((end_date IS NULL) OR (NEW.match_date
BETWEEN start_date AND end_date)));
              seekerid= (SELECT players.peopleid
                    FROM players right outer join plays_for on players.peopleid=plays_for.peopleid right outer
join positions on positions.positionid=plays_for.positionid
                    WHERE name='seeker' AND teamid=my_teamid AND end_date IS NULL);


            IF ((SELECT teamid from plays where match_date=NEW.match_date ORDER BY teamid DESC LIMIT 1) <>
my_teamID AND (SELECT teamid from plays where match_date=NEW.match_date ORDER BY teamid ASC LIMIT 1) <>
my_teamID) OR ((SELECT teamid from plays where match_date=NEW.match_date ORDER BY teamid DESC LIMIT 1) IS NULL
AND (SELECT teamid from plays where match_date=NEW.match_date ORDER BY teamid ASC LIMIT 1) IS NULL) THEN
                        INSERT INTO plays(match_date, teamID, totalpts, caught_snitch)
                              VALUES(NEW.match_date,my_teamID,NEW.pts_earned,snitch);
            END IF;

            UPDATE plays SET totalpts=totalpts+NEW.pts_earned WHERE teamid=my_teamID AND
match_date=NEW.match_date;

            IF (seekerid=NEW.peopleID) AND (NEW.pts_earned>0) THEN
                        UPDATE plays SET caught_snitch=true WHERE teamid=my_teamID AND
```

```
match_date=NEW.match_date;
                  END IF;

      RETURN NEW;
    END;
$to_plays$ LANGUAGE plpgsql;
```

plays_to_cup - This function returns a trigger and takes all of the data that is being input into plays (via the previous stored procedure) and populates quidditch_cup.

```
CREATE OR REPLACE FUNCTION plays_to_cup() RETURNS trigger AS $to_cup$
    DECLARE
      my_start_date int;
      my_end_date   int;
    BEGIN
      IF ((date_part('month', NEW.match_date) >= 9) AND (date_part('month', NEW.match_date) <> 12)) THEN
            my_start_date = date_part('year', NEW.match_date);
            my_end_date = date_part('year', NEW.match_date)+1;
      ELSIF date_part('month', NEW.match_date) < 6 THEN
            my_end_date = date_part('year', NEW.match_date);
            my_start_date = date_part('year', NEW.match_date)-1;
      ELSE
            RETURN NEW;
      END IF;

      IF (SELECT teamid from quidditch_cup WHERE season_start=my_start_date AND teamid=NEW.teamid) IS NULL
THEN
            INSERT INTO quidditch_cup(season_start, season_end, teamID, totalpts)
                  VALUES(my_start_date, my_end_date, NEW.teamid, 0);
      END IF;
      RETURN NEW;
    END;
$to_cup$ LANGUAGE plpgsql;
```

match_to_ref - This function returns a trigger and increments the games refereed row in referees depending on which ref was working which match.

```
CREATE OR REPLACE FUNCTION matchtoref() RETURNS trigger AS $m2r$
    BEGIN
          UPDATE referees SET games_refereed=games_refereed+1 WHERE peopleid=NEW.peopleid;
    RETURN NEW;
    END;
$m2r$ LANGUAGE plpgsql;
```

update_player_stats - This function returns a trigger and updates / adds to each player's stats (in the players table) according to the stats they received in the last game.

```
CREATE OR REPLACE FUNCTION update_player_stats() RETURNS trigger AS $update_player_stats$
    BEGIN
           -- if the player has not yet played a game as a starter they will have NULL stats, this checks to
see if they have NULL stats and are on the starting roster
           IF (select career_pts_earned from players where peopleID=NEW.PeopleID) IS NULL THEN
                 IF NEW.did_play THEN
                       -- if the check passes it then initalizes the player with all 0s for stats. fun!
                       UPDATE players
                       SET    career_pts_earned=0, career_penalties_recieved=0,
career_games_played_as_starter=0, season_pts_earned=0, season_penalties_recieved=0,
season_games_played_as_starter=0
                       WHERE  players.peopleID=NEW.peopleID;
                 END IF;
           END IF;
           -- the bread and butter
           UPDATE players
           SET    career_pts_earned=career_pts_earned+NEW.pts_earned,
career_penalties_recieved=career_penalties_recieved+NEW.penalties_recieved,
career_games_played_as_starter=career_games_played_as_starter+1,
season_pts_earned=season_pts_earned+NEW.pts_earned,
```

```
season_penalties_recieved=season_penalties_recieved+NEW.penalties_recieved,
season_games_played_as_starter=season_games_played_as_starter+1
            WHERE  players.peopleID=NEW.peopleID;
      RETURN NEW;
    END;
$update_player_stats$ LANGUAGE plpgsql;
```

add_data_to_cup - This function returns a trigger and updates the finalized total points for each team in the quidditch_cup table. NOTE: this only performs an update to the most recent season

```
CREATE OR REPLACE FUNCTION add_data_to_cup() RETURNS trigger AS $data_to_cup$
    DECLARE
      rec     record;
      cnt     int;
      temppts int;
      tempid  int;
      sdate   date;
      edate   date;
    BEGIN

            sdate = '9-1-' || (select season_start from quidditch_cup order by season_start DESC limit 1);
            edate = '6-1-' || (select season_end from quidditch_cup order by season_end DESC limit 1);
            temppts=0;
            cnt=1;
            while (cnt <5) LOOP
                  update quidditch_cup set totalpts = (SELECT SUM(totalpts) FROM plays WHERE teamID=cnt AND
match_date BETWEEN sdate AND edate) WHERE season_start=(select season_start from quidditch_cup order by
season_start desc limit 1) AND season_end=(select season_end from quidditch_cup order by season_end desc limit
1) AND teamid=cnt;
                  cnt = cnt+1;

            END LOOP;
            cnt=1;

            RETURN NEW;
    END;
$data_to_cup$ LANGUAGE plpgsql;
```

Triggers:

data_to_cup - This trigger executes the add_data_to_cup function (See: Stored Procedures) after data has been inserted into quidditch cup.

```
CREATE TRIGGER data_to_cup AFTER INSERT ON quidditch_cup
    FOR EACH ROW EXECUTE PROCEDURE add_data_to_cup();
```

	season_start integer	season_end integer	teamid integer	totalpts integer
1	2015	2016	1	0
2	2015	2016	4	0
3	2015	2016	2	0
4	2015	2016	3	0
5	1995	1996	1	0
6	1994	1995	1	0
7	1988	1989	1	0
8	1974	1975	1	0
9	1996	1997	1	0
10	1992	1993	4	0
11	1993	1994	2	0
12	1974	1975	4	0

before

	season_start integer	season_end integer	teamid integer	totalpts integer
1	1995	1996	1	0
2	1994	1995	1	0
3	1988	1989	1	0
4	1974	1975	1	0
5	1996	1997	1	0
6	1992	1993	4	0
7	1993	1994	2	0
8	1974	1975	4	0
9	2015	2016	1	1780
10	2015	2016	2	3240
11	2015	2016	3	3980
12	2015	2016	4	600

after

update_player_stats - This trigger executes the update_player_stats function (See: Stored Procedures) before data has been inserted into game_stats. NOTE: There are more than 15 rows, this is all the doc could display

```
CREATE TRIGGER update_player_stats BEFORE INSERT ON game_stats
    FOR EACH ROW EXECUTE PROCEDURE update_player_stats();
```

	peopleid integer	career_pts_earned integer	career_penalties_recieved integer	career_games_played_as_starter integer	season_pts_earned integer	season_penalties_recieved integer	season_games_played_as_starter integer
1	1						
2	4						
3	5						
4	6						
5	7						
6	8						
7	9						
8	10						
9	11						
10	12						
11	13						
12	14						
13	15						
14	16						
15	17						

before

	peopleid integer	career_pts_earned integer	career_penalties_recieved integer	career_games_played_as_starter integer	season_pts_earned integer	season_penalties_recieved integer	season_games_played_as_starter integer
1	1	0	1	3	0	1	3
2	4	300	0	3	300	0	3
3	5	0	6	3	0	6	3
4	6	10	3	3	10	3	3
5	7	0	2	1	0	2	1
6	8	0	9	1	0	9	1
7	9	0	10	1	0	10	1
8	10	150	0	1	150	0	1
9	11	150	0	1	150	0	1
10	12	0	0	1	0	0	1
11	13	210	5	1	210	5	1
12	14	180	0	1	180	0	1
13	15	0	2	3	0	2	3
14	16	0	2	3	0	2	3
15	17	0	1	3	0	1	3

after

m2r - This trigger executes the matchtoref function (See: Stored Procedures) before data has been inserted into matches.

```
CREATE TRIGGER m2r BEFORE INSERT ON matches
    FOR EACH ROW EXECUTE PROCEDURE matchtoref();
```

	peopleid integer	games_refereed integer
1	2	42
2	3	251

before

	peopleid integer	games_refereed integer
1	2	42
2	3	265

after

plays_to_cup - This trigger executes the plays_to_cup function (See: Stored Procedures) after data in matches has been updated. NOTE: totalpts column in after is already populated due to the data_to_cup function triggering.

```
CREATE TRIGGER plays_to_cup AFTER UPDATE ON plays
    FOR EACH ROW EXECUTE PROCEDURE plays_to_cup();
```

	season_start integer	season_end integer	teamid integer	totalpts integer

before

	season_start integer	season_end integer	teamid integer	totalpts integer
1	1995	1996	1	0
2	1994	1995	1	0
3	1988	1989	1	0
4	1974	1975	1	0
5	1996	1997	1	0
6	1992	1993	4	0
7	1993	1994	2	0
8	1974	1975	4	0
9	2015	2016	1	1780
10	2015	2016	2	3240
11	2015	2016	3	3980
12	2015	2016	4	600

after

plays_to_cup - This trigger executes the gamedata_to_plays function (See: Stored Procedures) after data has been inserted into matches. NOTE: There are more than 15 rows, this is all the doc could display

```
CREATE TRIGGER gamedata_to_plays AFTER INSERT ON game_stats
    FOR EACH ROW EXECUTE PROCEDURE gamedata_to_plays();
```

	match_date date	teamid integer	totalpts integer	caught_snitch boolean

before

	match_date date	teamid integer	totalpts integer	caught_snitch boolean
1	2015-11-07	1	360	t
2	2015-11-07	4	300	t
3	2015-11-28	2	1390	f
4	2015-11-28	3	1540	t
5	2016-02-20	3	1520	t
6	2016-02-20	4	150	f
7	2016-03-12	1	940	t
8	2016-03-12	2	1540	f
9	2016-05-07	2	310	f
10	2016-05-07	4	150	t
11	2016-05-28	1	480	t
12	2016-05-28	3	920	f
13	1995-02-04	1	0	f
14	1996-05-25	1	150	f
15	1988-11-05	1	300	f

after

Roles:

Administrator Role - The database administrator, full, unadulterated access.

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

Headmaster Role - The only non-directly involved faculty member allowed to edit the database. Essentially root.

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

Referee Role - It is the referee's job to input new players, new referees, new matches, new brooms, and new game data in general.

```
create role referee;
revoke all on all tables in schema public from referee;
grant select on all tables in schema public to referee;
grant insert on people, players, referees, matches, game_stats, flys, brooms, plays_for, balls_in_match to
referee;
grant update on people, players, referees, matches, game_stats, flys, brooms, plays_for, balls_in_match,
quidditch_cup, teams to referee;
```

Student Role - The student may query the database to learn about their classmates or conduct research for a project / strategic planning.

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

Implementation Notes:

When tasked with creating the HIHQCDB there was almost no data remaining from the previous system. Thus the only data within the system right now is the most recent season of the Inter-House Quidditch Cup and some saved data that was donated by specific house quidditch historical clubs.

Due to the nature of Quidditch and specifically, this cup's scoring system, Win/loss/tie was not needed. The way the HIHQC determines a winner is just through which team has accumulated the most points over the course of the season.

balls_in_match was created to ensure the proper inclusion of the balls table in the database, and for show matches where there could perhaps be 4 teams pitted against each other in 2v2 fashion with extra balls thrown in there for fun.

Although there was no lack effort, there were some pieces of information that were just impossible to accurately obtain. ie. the weights and speeds of the different balls. After scouring anything that could be possibly canon (HP Series, Cursed Child, Movies, Quidditch Through The Ages) the remaining data was determined via the EU (expanded universe), scifi stackexchange, fan theory forums, and myself. Some numbers, such as the weight of a bludger, was calculated using information directly from the books and yet still the numbers still came up absurdly wrong. A solid iron ball that weighs 66KG and has an avg speed of 97KPH would bore holes through these children competing. My conclusion when presented with this data? Magic.

Known Problems / Future Enhancements:

- This database as of now works best when moving forward. Inputting data from previous years through game_stats will trigger all of the functions but some functions are designed to utilize the most recent year and continue.

- There is no way to easily grab win/loss/tie values.

- I would like to eventually add a way to convert knut values into galleon,sickle,knut via a stored procedure.

- There are other tables that could be added and useful to the database, such as an injuries table for people or a set of underground gambling table.

- Perhaps adding a punishment to a certain number of penalties in a game? In a season?