Designing a Database

We will be working with a file of Major League Baseball games from Retrosheet. Retrosheet compiles detailed statistics on baseball games from the 1800s through to today. The main file we will be working from, game_log.csv, has been produced by combining 127 separate CSV files from retrosheet, and has been pre-cleaned to remove some inconsistencies. The game log has hundreds of data points on each game which we will normalize into several separate tables using SQL, providing a robust database of game-level statistics.

Load and explore the data

In [1]:

```
#import and set pandas to not truncate rows/cols
import pandas as pd
pd.set_option('max_columns', 180)
pd.set_option('max_rows', 200000)
pd.set_option('max_colwidth', 5000)
```

game log data

In [2]:

```
#read game log
game_log = pd.read_csv('game_log.csv')
```

```
/dataquest/system/env/python3/lib/python3.4/site-packages/IPython/
core/interactiveshell.py:2723: DtypeWarning: Columns (12,13,14,15,
19,20,81,82,83,84,85,86,87,88,93,94,95,96,97,98,99,100,105,106,108
,109,111,112,114,115,117,118,120,121,123,124,126,127,129,130,132,1
33,135,136,138,139,141,142,144,145,147,148,150,151,153,154,156,157
,160) have mixed types. Specify dtype option on import or set low_
memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [3]:

```
game_log.head()
```

Out[3]:

	date	number_of_game	day_of_week	v_name	v_league	v_game_number	h_name
0	18710504	0	Thu	CL1	NaN	1	FW1
1	18710505	0	Fri	BS1	NaN	1	WS3
2	18710506	0	Sat	CL1	NaN	2	RC1
3	18710508	0	Mon	CL1	NaN	3	CH1
4	18710509	0	Tue	BS1	NaN	2	TRO

In [4]:

```
game_log.shape
```

Out[4]:

```
(171907, 161)
```

In [5]:

```
game_log.tail()
```

Out[5]:

	date	number_of_game	day_of_week	v_name	v_league	v_game_number	h_r
171902	20161002	0	Sun	MIL	NL	162	
171903	20161002	0	Sun	NYN	NL	162	
171904	20161002	0	Sun	LAN	NL	162	
171905	20161002	0	Sun	PIT	NL	162	
171906	20161002	0	Sun	MIA	NL	161	

game_log is huge, When creating our database it will not be necessary to include all of this information.
Data appears to be ordered by 'date' and shows information about one game per row
What is contained in the data

- date, time, day of week
- the score, number of homeruns etc
- the umpires of the game, attendance and game length
- information about the visiting team and the home team
- both visiting and home teams starting line up in the order they batted and their respective defensive position (1-9)
- stats such as winning pitcher, losing pitcher

Park codes

In [6]:

```
park_codes = pd.read_csv('park_codes.csv')
```

In [7]:

```
park_codes.shape
```

Out[7]:

```
(252, 9)
```

In [8]:

```
park_codes.head()
```

Out[8]:

	park_id	name	aka	city	state	start	end	league	
0	ALB01	Riverside Park	NaN	Albany	NY	09/11/1880	05/30/1882	NL	TRN:9/11
1	ALT01	Columbia Park	NaN	Altoona	PA	04/30/1884	05/31/1884	UA	
2	ANA01	Angel Stadium of Anaheim	Edison Field; Anaheim Stadium	Anaheim	CA	04/19/1966	NaN	AL	
3	ARL01	Arlington Stadium	NaN	Arlington	TX	04/21/1972	10/03/1993	AL	
4	ARL02	Rangers Ballpark in Arlington	The Ballpark in Arlington; Ameriquest Fl	Arlington	TX	04/11/1994	NaN	AL	

park_codes is simply an dataframe consisting of information about stadiums and baseball venues.
It doesn't take much to notice that the park_id column in park_codes AND game_log contain the same values, meaning this could be the foreign key connecting the two tables.
also included are

- its location
- the league of the matches played there
- The opening and closing dates (if applicable) of the stadium

Person codes

In [9]:

```
person_codes = pd.read_csv('person_codes.csv')
```

In [10]:

```
person_codes.shape
```

Out[10]:

```
(20494, 7)
```

In [11]:

```
person_codes.head()
```

Out[11]:

	id	last	first	player_debut	mgr_debut	coach_debut	ump_debut
0	aardd001	Aardsma	David	04/06/2004	NaN	NaN	NaN
1	aaroh101	Aaron	Hank	04/13/1954	NaN	NaN	NaN
2	aarot101	Aaron	Tommie	04/10/1962	NaN	04/06/1979	NaN
3	aased001	Aase	Don	07/26/1977	NaN	NaN	NaN
4	abada001	Abad	Andy	09/10/2001	NaN	NaN	NaN

person_codes contains data about each player, coach, manager and umpire. First/last names and their debut date are all included.

- The person_codes 'id' column could be used as a foreign key for game*log (v/h)* (player/coach/umpire)_id columns
- The concatenation of both first and last name could be the foreign key to game*log (v/h)* (player/umpire/coach)_name columns
- player debut is relatable to the 'date' column in game_logs after some cleaning

Team codes

In [12]:

```
team_codes = pd.read_csv('team_codes.csv')
```

In [13]:

```
team_codes.shape
```

Out[13]:

```
(150, 8)
```

In [14]:

```
team_codes.head()
```

Out[14]:

	team_id	league	start	end	city	nickname	franch_id	seq
0	ALT	UA	1884	1884	Altoona	Mountain Cities	ALT	1
1	ARI	NL	1998	0	Arizona	Diamondbacks	ARI	1
2	BFN	NL	1879	1885	Buffalo	Bisons	BFN	1
3	BFP	PL	1890	1890	Buffalo	Bisons	BFP	1
4	BL1	NaN	1872	1874	Baltimore	Canaries	BL1	1

Similarly team_codes contains info about each team within the data such as name, league and their established dates. Here are a few standout points from first glance

- franch_id looks to be identical to team_id
- team_id corresponds to v_name and h_name in game_log

In [15]:

```
# lets look at franch_id
team_codes['franch_id'].value_counts(dropna=False).head()
```

Out[15]:

```
BS1     4
SE1     3
LAA     3
TRN     3
MLA     3
Name: franch_id, dtype: int64
```

In [16]:

```
#and the team_id
team_codes['team_id'].value_counts(dropna=False,sort=True).head()
```

Out[16]:

```
MIL    2
ANA    1
SE1    1
CHF    1
BR3    1
Name: team_id, dtype: int64
```

They're clearly not the same, lets investigate

In [17]:

```
# try looking at the franch_id BS1 which appears in 4 rows
team_codes.loc[team_codes['franch_id']=='BS1', :]
```

Out[17]:

	team_id	league	start	end	city	nickname	franch_id	seq
21	BS1	NaN	1871	1875	Boston	Braves	BS1	1
22	BSN	NL	1876	1952	Boston	Braves	BS1	2
23	MLN	NL	1953	1965	Milwaukee	Braves	BS1	3
24	ATL	NL	1966	0	Atlanta	Braves	BS1	4

So having researched the above teams, they are the same franchise which has been renamed and relocated on numerous occasions.
'The Braves were founded in Boston, Massachusetts, in 1871 then, in 1953, the team moved to Milwaukee, Wisconsin, and became the Milwaukee Braves, followed by the final move to Atlanta in 1966' - wikipedia

Essentially the franchise and team name/id are different. This may be an important factor to consider later on

In [18]:

```
#lets also look at the only team who's id appears more than once in team_codes
team_codes.loc[team_codes['team_id']=='MIL', :]
```

Out[18]:

	team_id	league	start	end	city	nickname	franch_id	seq
112	MIL	AL	1970	1997	Milwaukee	Brewers	SE1	2
113	MIL	NL	1998	0	Milwaukee	Brewers	SE1	3

investigating the brewers revealed the following
'In 1998, the Brewers changed leagues, going from the American League to the National League. They were put in the then recently created NL Central.' - wikipedia

consisdering the Brewers are the only team with multiple appearances within team_codes, it would appear they are the only team to have changed leagues

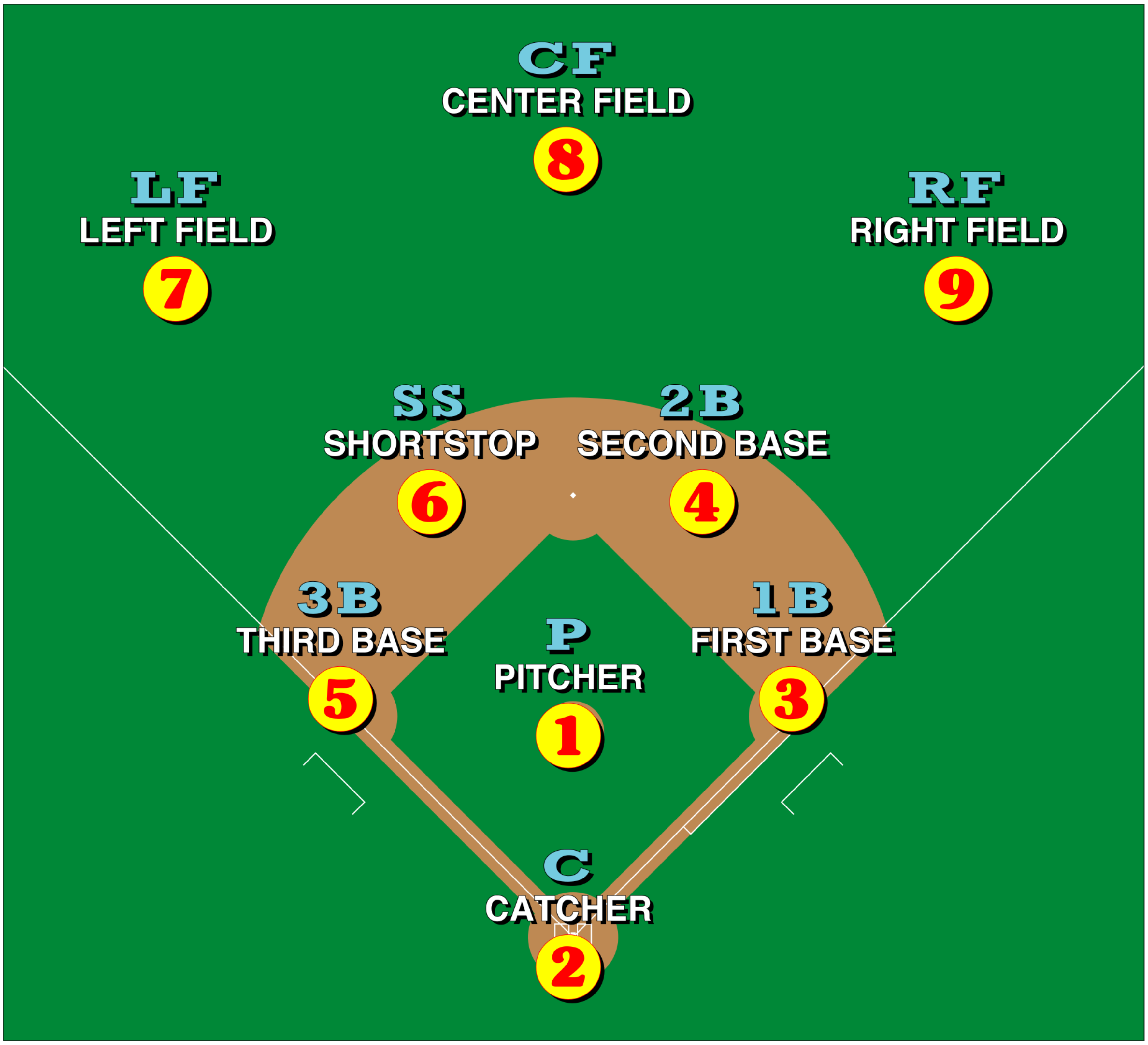
Understanding the data

Myself not being particularly well versed in baseball rules and regulations I figure its a good idea for me to clear up any uncertancies I have regarding the meaning behind certain aspects of the data. Better to do this now than later.

The aim of the game

- In baseball, each team field 9 players, whom in each half inning either 'bat' or 'field'.
- One inning consists of both teams having a turn at batting, half an inning consists of one team batting
- The aim is to score as many runs as possible whilst in bat and also reduce the amount of runs for the batting team whilst in field.
- A game consists of 9 innings, if the game is tied at the end, extra innings are played to resolve the contest.
- The team in the field attempts to prevent runs from scoring by recording outs, which remove opposing players from offensive action until their turn in their team's batting order comes up again.
- The players on the team at bat attempt to score runs by circling or completing a tour of the four bases set at the corners of the square-shaped baseball diamond.
- Each player take turns playing offense (batting and baserunning) and defense (pitching and fielding).

Defensive positions



The picture is pretty self explanatory, each player is given a number relating to their position when their team is fielding

The Leagues

In [19]:

```
team_codes['league'].value_counts(dropna=False)
```

Out[19]:

```
NL     45
NaN    26
AL     25
AA     24
UA     13
FL      9
PL      8
Name: league, dtype: int64
```

From the source of the data, I found the following

The league codes are:
NA = National Association
NL = National League
AA = American Association
UA = Union Association
PL = Players League
AA = American League
FL = Federal League

In [20]:

```
# what does 'number_of_game' mean
game_log['number_of_game'].value_counts()
```

Out[20]:

```
0    142010
2     14947
1     14947
3         3
Name: number_of_game, dtype: int64
```

from the souce
"0" -- a single game
"1" -- the first game of a double (or triple) header including seperate admission doubleheaders
"2" -- the second game of a double (or triple) header including seperate admission doubleheaders
"3" -- the third game of a triple-header
"A" -- the first game of a double-header involving 3 teams
"B" -- the second game of a double-header involving 3 teams

Create a Database

In [21]:

```
import sqlite3 as sql
#command, query, show_table helpers
DB='mlb.db'
def run_query(q):
    with sql.connect(DB) as conn:
        return pd.read_sql(q,conn)

def run_command(c):
    with sql.connect(DB) as conn:
        conn.execute('PRAGMA foreign_keys = ON;')
        conn.isolation_level=None
        conn.execute(c)
def show_tables():
    query = '''
             SELECT
             name,
             type
             FROM
             sqlite_master
             WHERE type IN ("table","view");
             '''
    return run_query(query)
```

In [22]:

```
tables = {'person_codes':person_codes,
          'game_log':game_log,
          'park_codes':park_codes,
          'team_codes':team_codes}
```

In [23]:

```
for name, data in tables.items():
    with sql.connect(DB) as conn:
        conn.execute('DROP TABLE IF EXISTS {};'.format(name))
        data.to_sql(name, conn, index=False)
```

In [24]:

```
#check it worked
run_query('SELECT * FROM game_log LIMIT 5')
```

Out[24]:

```
       date  number_of_game  day_of_week  v_name  v_league  v_game_number  h_name
0  18710504               0          Thu     CL1      None              1     FW1
1  18710505               0          Fri     BS1      None              1     WS3
2  18710506               0          Sat     CL1      None              2     RC1
3  18710508               0          Mon     CL1      None              3     CH1
4  18710509               0          Tue     BS1      None              2     TRO
```

In [25]:

```
# create a game_id column to act as the primary key
command = '''
          ALTER TABLE game_log
          ADD game_id TEXT;

'''
run_command(command)
```

In [26]:

```
# concatenate date,home_name,number_of_game and assign the game_id as the comp
ound primary key
# This idea is taken from the EVENT files from the data source, which contains
info about the events in each game and  uses the same primary key to
# identify the game
command = '''
          UPDATE game_log
          SET game_id = date || h_name || number_of_game
          WHERE game_id IS NULL'''
run_command(command)
```

In [27]:

```
#check it was made, look at the last column
run_query('SELECT * FROM game_log LIMIT 5')
```

Out[27]:

```
        date  number_of_game  day_of_week  v_name  v_league  v_game_number  h_name
0   18710504               0          Thu     CL1      None              1     FW1
1   18710505               0          Fri     BS1      None              1     WS3
2   18710506               0          Sat     CL1      None              2     RC1
3   18710508               0          Mon     CL1      None              3     CH1
4   18710509               0          Tue     BS1      None              2     TRO
```

Opportunities to normalize the data

From game_log, players/umpires/manager name is not an attribute of game_id(the primary key) but an attribute of player/umpire/manager_id. Therefore we have an non-key field functionally dependant upon another non-key field. Since the person name can be found in person_codes, we can sadely remove person_name fields from game_log to prevent duplication

We can also safely remove h_league and v_league from game_log for the same reasons, they can be found in team_codes

There are some other things worth noticing

- 'v_at_bats' up to 'v_triple_plays' contain statistics for the visiting team in each game. The same columns are then repeated for home team. creating a new table with generic columns and (game_id + home/away) as a primary compound key makes sense here and to have two lines per game_id, one for the home team and the other for the visitors
- A seperate table could be made for each players defensive and offensive positions for each appearance in a given game with (game_id + player_id) as the compound primary key.
- could do the same for umpires and managers
- There are other stats such as winning_pitcher, losing_pitcher, winning_rbi_batter_id which could be combined into a new table containing only awards/shame statistics about a single game_id

In [28]:

```
#simplify people_codes
run_query('select * from person_codes limit 5')
```

Out[28]:

	id	last	first	player_debut	mgr_debut	coach_debut	ump_debut
0	aardd001	Aardsma	David	04/06/2004	None	None	None
1	aaroh101	Aaron	Hank	04/13/1954	None	None	None
2	aarot101	Aaron	Tommie	04/10/1962	None	04/06/1979	None
3	aased001	Aase	Don	07/26/1977	None	None	None
4	abada001	Abad	Andy	09/10/2001	None	None	None

We can remove the debut columns since this information can be determined from the game_log file if needs be, say we wanted to find the debut date of Hank Aaron, we could simply select all games in which he was present and then sort by date.

In [29]:

```
#simplify team_codes
run_query('select * from team_codes limit 5')
```

Out[29]:

	team_id	league	start	end	city	nickname	franch_id	seq
0	ALT	UA	1884	1884	Altoona	Mountain Cities	ALT	1
1	ARI	NL	1998	0	Arizona	Diamondbacks	ARI	1
2	BFN	NL	1879	1885	Buffalo	Bisons	BFN	1
3	BFP	PL	1890	1890	Buffalo	Bisons	BFP	1
4	BL1	None	1872	1874	Baltimore	Canaries	BL1	1

league can be removed since we can get that in game_log, and so can we obtain the start and end dates of the teams from game_log

In [30]:

```
#simplify park_codes
run_query('select * from park_codes limit 5')
```

Out[30]:

	park_id	name	aka	city	state	start	end	league	
0	ALB01	Riverside Park	None	Albany	NY	09/11/1880	05/30/1882	NL	TRN:9,
1	ALT01	Columbia Park	None	Altoona	PA	04/30/1884	05/31/1884	UA	
2	ANA01	Angel Stadium of Anaheim	Edison Field; Anaheim Stadium	Anaheim	CA	04/19/1966	None	AL	
3	ARL01	Arlington Stadium	None	Arlington	TX	04/21/1972	10/03/1993	AL	
4	ARL02	Rangers Ballpark in Arlington	The Ballpark in Arlington; Ameriquest Fl	Arlington	TX	04/11/1994	None	AL	

start and end dates can be removed like for team_codes, as can league

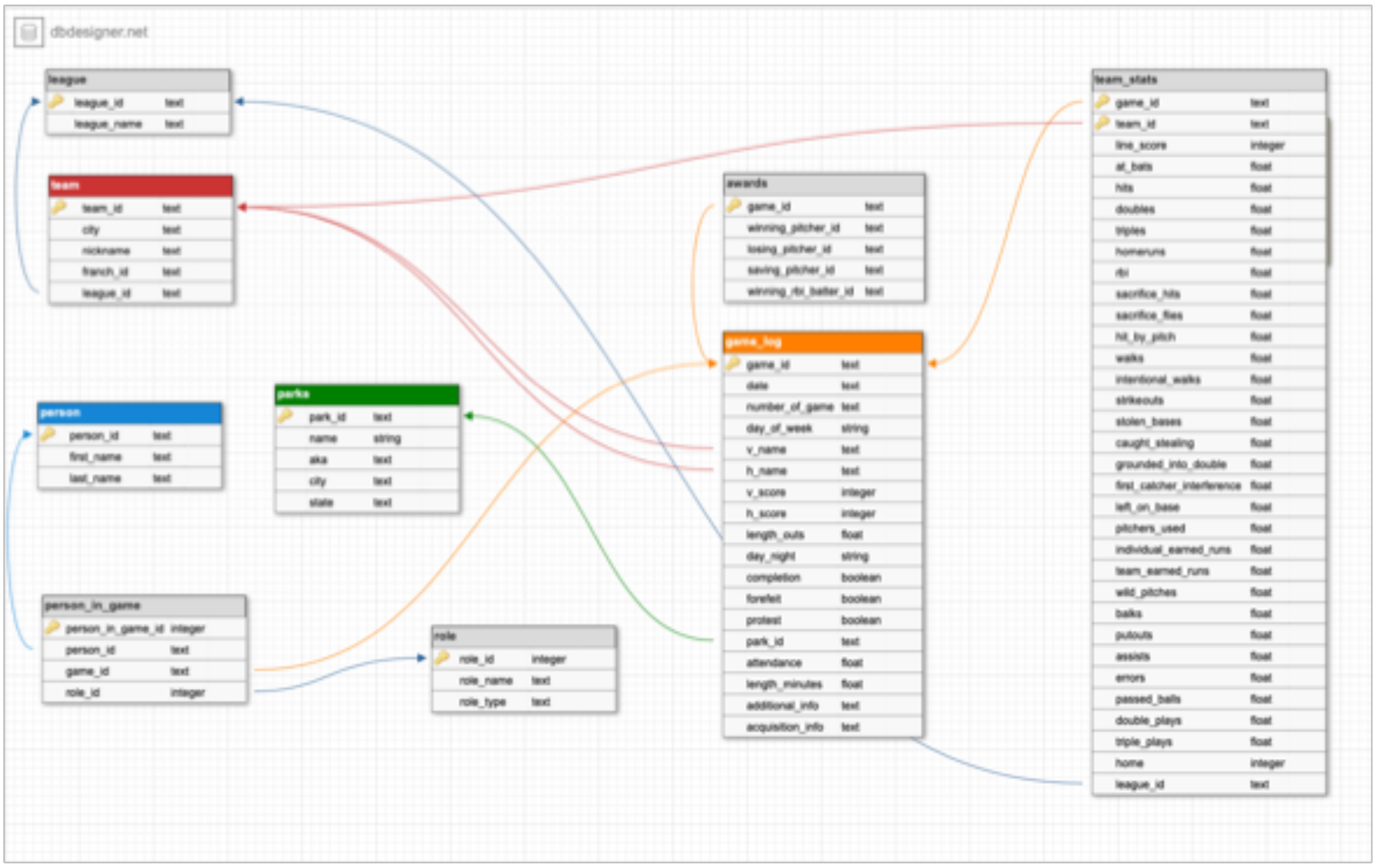
In [31]:

```
#just making sure i've understood the defensive positions, def_pos = 1 corresp
onds to starting pitcher
run_query('select * from game_log where v_player_2_def_pos = 1 limit 5')
```

Out[31]:

	date	number_of_game	day_of_week	v_name	v_league	v_game_number	h_name
0	18710520	0	Sat	PH1	None	1	BS1
1	18710526	0	Fri	FW1	None	5	CL1
2	18710617	0	Sat	PH1	None	6	WS3
3	18710619	0	Mon	FW1	None	6	TRO
4	18710621	0	Wed	FW1	None	7	BS1

Designing a schema

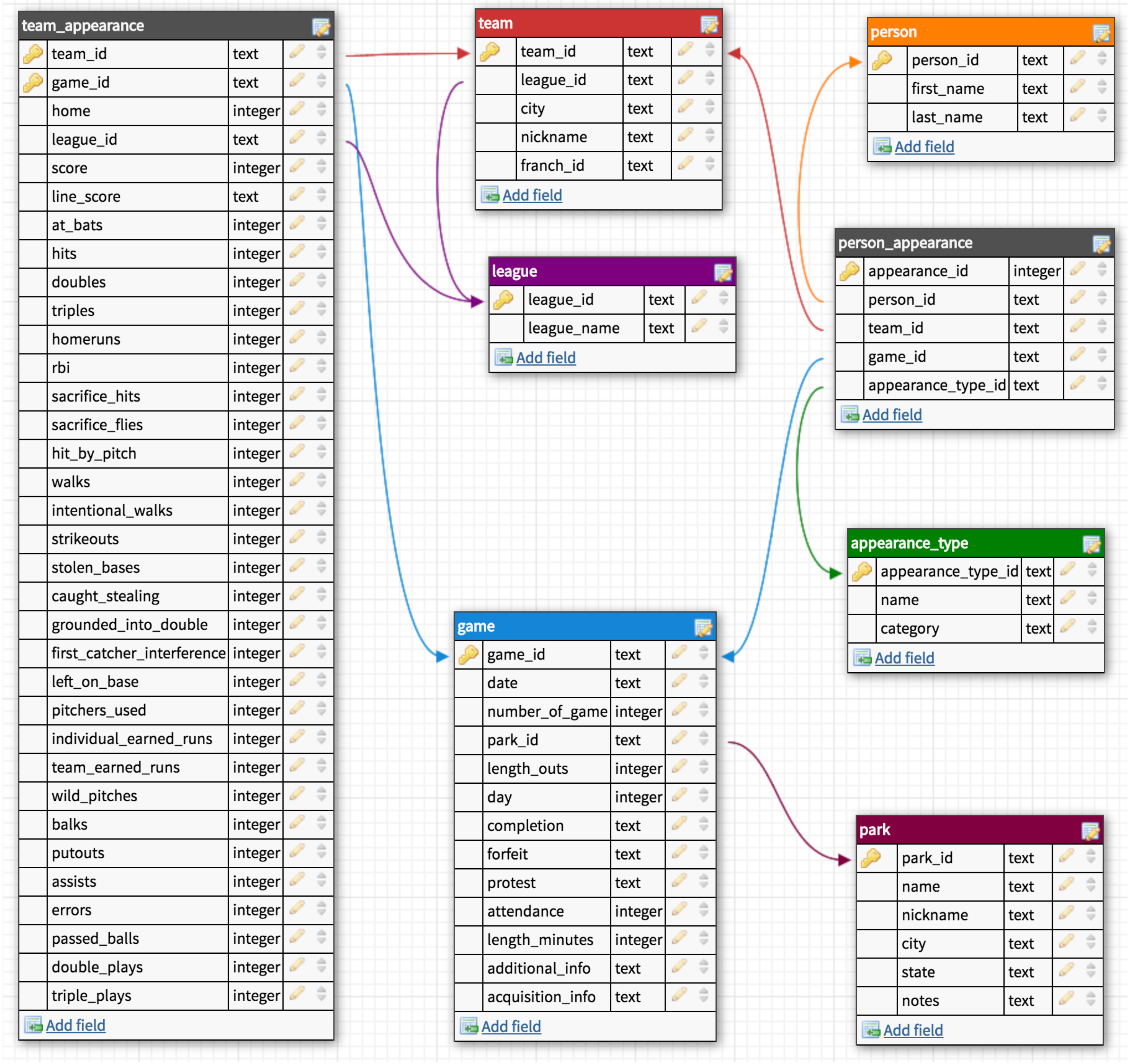


My attempt at designing the schema

Making the tables

Start with tables which do not have a foreign key and so can exist straight away

The correct schema was provided by dataquest and will be used from here on in. noticeable changes include the awards table being removed in favour of placing the awards information in the role table.
day_night has also been changed to just 'day' which will be a boolean value



team_appearance

🔑	team_id	text
🔑	game_id	text
	home	integer
	league_id	text
	score	integer
	line_score	text
	at_bats	integer
	hits	integer
	doubles	integer
	triples	integer
	homeruns	integer
	rbi	integer
	sacrifice_hits	integer
	sacrifice_flies	integer
	hit_by_pitch	integer
	walks	integer
	intentional_walks	integer
	strikeouts	integer
	stolen_bases	integer
	caught_stealing	integer
	grounded_into_double	integer
	first_catcher_interference	integer
	left_on_base	integer
	pitchers_used	integer
	individual_earned_runs	integer
	team_earned_runs	integer
	wild_pitches	integer
	balks	integer
	putouts	integer
	assists	integer
	errors	integer
	passed_balls	integer
	double_plays	integer
	triple_plays	integer

Add field

team

🔑	team_id	text
	league_id	text
	city	text
	nickname	text
	franch_id	text

Add field

league

🔑	league_id	text
	league_name	text

Add field

person

🔑	person_id	text
	first_name	text
	last_name	text

Add field

person_appearance

🔑	appearance_id	integer
	person_id	text
	team_id	text
	game_id	text
	appearance_type_id	text

Add field

appearance_type

🔑	appearance_type_id	text
	name	text
	category	text

Add field

game

🔑	game_id	text
	date	text
	number_of_game	integer
	park_id	text
	length_outs	integer
	day	integer
	completion	text
	forfeit	text
	protest	text
	attendance	integer
	length_minutes	integer
	additional_info	text
	acquisition_info	text

Add field

park

🔑	park_id	text
	name	text
	nickname	text
	city	text
	state	text
	notes	text

Add field

tables with no foreign keys

In [32]:

```
# start with Person
#create empty table
create = '''
          CREATE TABLE IF NOT EXISTS person(
          person_id TEXT PRIMARY KEY,
          first_name TEXT,
          last_name TEXT
          );
'''

run_command(create)
#fill table
fill = '''
       INSERT OR IGNORE INTO person
       SELECT
       id,
       first,
       last
       FROM person_codes
       ;
        '''
run_command(fill)
```

In [33]:

```
run_query('select * from person limit 3')
```

Out[33]:

	person_id	first_name	last_name
0	aardd001	David	Aardsma
1	aaroh101	Hank	Aaron
2	aarot101	Tommie	Aaron

In [34]:

```
#park table

create = '''
          CREATE TABLE IF NOT EXISTS park(
          park_id TEXT PRIMARY KEY,
          name TEXT,
          aka TEXT,
          city TEXT,
          state TEXT,
          notes TEXT
          );
'''

run_command(create)
#fill table
fill = '''
       INSERT OR IGNORE INTO park
       SELECT
       park_id,
       name,
       aka,
       city,
       state,
       notes
       FROM park_codes
       ;
        '''
run_command(fill)
run_query('select * from park limit 3')
```

Out[34]:

	park_id	name	aka	city	state	notes
0	ALB01	Riverside Park	None	Albany	NY	TRN:9/11/80;6/15&9/10/1881;5/16-5/18&5/30/1882
1	ALT01	Columbia Park	None	Altoona	PA	None
2	ANA01	Angel Stadium of Anaheim	Edison Field; Anaheim Stadium	Anaheim	CA	None

In [35]:

```
# league
create = '''
          CREATE TABLE IF NOT EXISTS league(
          league_id TEXT PRIMARY KEY,
          league_name TEXT
          );
'''

run_command(create)
#fill table
fill = '''
       INSERT OR IGNORE INTO league
       VALUES
       ('AA', 'American Association'),
       ('AL', 'American League'),
       ('NL', 'National League'),
       ('FL', 'Federal League'),
       ('PL', 'Players League'),
       ('UA', 'Union Association')
        '''
run_command(fill)
```

In [36]:

```
run_query('select * from league limit 5')
```

Out[36]:

	league_id	league_name
0	AA	American Association
1	AL	American League
2	NL	National League
3	FL	Federal League
4	PL	Players League

In [37]:

```
#role, make use of appearance_type.csv provided

roles = pd.read_csv('appearance_type.csv')
roles.rename(columns={'appearance_type_id': 'role_id'}, inplace=True)
# create
create = '''
          CREATE TABLE IF NOT EXISTS role(
          role_id TEXT PRIMARY KEY,
          name TEXT,
          category TEXT
          );
'''

run_command(create)
#fill table
#created empty role table, now use dataframe 'roles' to fill it

with sql.connect('mlb.db') as conn:
    roles.to_sql('role', conn,
                           index=False, if_exists='append')
```

In [38]:

```
#did it load correctly?
run_query('select * from role limit 5')
```

Out[38]:

	role_id	name	category
0	O1	Batter 1	offense
1	O2	Batter 2	offense
2	O3	Batter 3	offense
3	O4	Batter 4	offense
4	O5	Batter 5	offense

tables with foreign keys

In [39]:

```
#team
#create table
create = '''
         CREATE TABLE IF NOT EXISTS team(
         team_id TEXT PRIMARY KEY,
         league_id TEXT,
         city TEXT,
         nickname TEXT,
         franch_id TEXT,
         FOREIGN KEY (league_id) REFERENCES league(league_id)
         );
'''
run_command(create)

#fill table
fill = '''
       INSERT OR IGNORE INTO team
       SELECT
       team_id,
       league,
       city,
       nickname,
       franch_id
       FROM team_codes
       ;
'''

run_command(fill)
```

In [40]:

```
run_query('select * from team limit 5')
```

Out[40]:

	team_id	league_id	city	nickname	franch_id
0	ALT	UA	Altoona	Mountain Cities	ALT
1	ARI	NL	Arizona	Diamondbacks	ARI
2	BFN	NL	Buffalo	Bisons	BFN
3	BFP	PL	Buffalo	Bisons	BFP
4	BL1	None	Baltimore	Canaries	BL1

In [41]:

```
#game table
#create

create = '''
         CREATE TABLE IF NOT EXISTS game(
         game_id TEXT PRIMARY KEY,
         date TEXT,
         number_of_game INTEGER,
         park_id TEXT,
         length_outs INTEGER,
         day BOOLEAN,
         completion TEXT,
         forefeit TEXT,
         protest TEXT,
         attendance INTEGER,
         length_mintutes INTEGER,
         additional_info TEXT,
         acquisition_info TEXT,
         FOREIGN KEY (park_id) REFERENCES park(park_id)
         )
         ;

'''

run_command(create)

#fill

fill = '''
       INSERT OR IGNORE INTO game
       SELECT
       game_id,
       date,
       number_of_game,
       park_id,
       length_outs,
       CASE
           WHEN day_night = "D" THEN 1
           WHEN day_night = "N" THEN 0
           ELSE NULL
           END AS day,
       completion,
       forefeit,
       protest,
       attendance,
       length_minutes,
       additional_info,
       acquisition_info
       FROM game_log
'''

run_command(fill)
```

In [42]:

```
run_query('select * from game limit 10')
```

Out[42]:

	game_id	date	number_of_game	park_id	length_outs	day	completion	fore
0	18710504FW10	18710504	0	FOR01	54	1	None	N
1	18710505WS30	18710505	0	WAS01	54	1	None	N
2	18710506RC10	18710506	0	RCK01	54	1	None	N
3	18710508CH10	18710508	0	CHI01	54	1	None	N
4	18710509TRO0	18710509	0	TRO01	54	1	None	N
5	18710511CL10	18710511	0	CLE01	48	1	None	
6	18710513CL10	18710513	0	CIN01	54	1	None	N
7	18710513FW10	18710513	0	FOR01	54	1	None	N
8	18710515FW10	18710515	0	FOR01	54	1	None	N
9	18710516BS10	18710516	0	BOS01	54	1	None	N

In [43]:

```
# make sure only 1 or 0
run_query('''
          SELECT
          day,
          COUNT(game_id)
          FROM
          game
          GROUP BY day
          ''')
```

Out[43]:

	day	COUNT(game_id)
0	NaN	31757
1	0.0	57426
2	1.0	82724

In [ ]:

```

```

In [44]:

```
# use this to copy needed column names and types
run_query('''
       select
       (SELECT sql FROM sqlite_master
       WHERE name = "game_log"
       AND type = "table")
''')
```

Out[44]:

	(SELECT sql FROM sqlite_master WHERE name = "game_log" AND type = "table")
0	CREATE TABLE "game_log" (\n"date" INTEGER,\n "number_of_game" INTEGER,\n "day_of_week" TEXT,\n "v_name" TEXT,\n "v_league" TEXT,\n "v_game_number" INTEGER,\n "h_name" TEXT,\n "h_league" TEXT,\n "h_game_number" INTEGER,\n "v_score" INTEGER,\n "h_score" INTEGER,\n "length_outs" REAL,\n "day_night" TEXT,\n "completion" TEXT,\n "forefeit" TEXT,\n "protest" TEXT,\n "park_id" TEXT,\n "attendance" REAL,\n "length_minutes" REAL,\n "v_line_score" TEXT,\n "h_line_score" TEXT,\n "v_at_bats" REAL,\n "v_hits" REAL,\n "v_doubles" REAL,\n "v_triples" REAL,\n "v_homeruns" REAL,\n "v_rbi" REAL,\n "v_sacrifice_hits" REAL,\n "v_sacrifice_flies" REAL,\n "v_hit_by_pitch" REAL,\n "v_walks" REAL,\n "v_intentional_walks" REAL,\n "v_strikeouts" REAL,\n "v_stolen_bases" REAL,\n "v_caught_stealing" REAL,\n "v_grounded_into_double" REAL,\n "v_first_catcher_interference" REAL,\n "v_left_on_base" REAL,\n "v_pitchers_used" REAL,\n "v_individual_earned_runs" REAL,\n "v_team_earned_runs" REAL,\n "v_wild_pitches" REAL,\n "v_balks" REAL,\n "v_putouts" REAL,\n "v_assists" REAL,\n "v_errors" REAL,\n "v_passed_balls" REAL,\n "v_double_plays" REAL,\n "v_triple_plays" REAL,\n "h_at_bats" REAL,\n "h_hits" REAL,\n "h_doubles" REAL,\n "h_triples" REAL,\n "h_homeruns" REAL,\n "h_rbi" REAL,\n "h_sacrifice_hits" REAL,\n "h_sacrifice_flies" REAL,\n "h_hit_by_pitch" REAL,\n "h_walks" REAL,\n "h_intentional_walks" REAL,\n "h_strikeouts" REAL,\n "h_stolen_bases" REAL,\n "h_caught_stealing" REAL,\n "h_grounded_into_double" REAL,\n "h_first_catcher_interference" REAL,\n "h_left_on_base" REAL,\n "h_pitchers_used" REAL,\n "h_individual_earned_runs" REAL,\n "h_team_earned_runs" REAL,\n "h_wild_pitches" REAL,\n "h_balks" REAL,\n "h_putouts" REAL,\n "h_assists" REAL,\n "h_errors" REAL,\n "h_passed_balls" REAL,\n "h_double_plays" REAL,\n "h_triple_plays" REAL,\n "hp_umpire_id" TEXT,\n "hp_umpire_name" TEXT,\n "1b_umpire_id" TEXT,\n "1b_umpire_name" TEXT,\n "2b_umpire_id" TEXT,\n "2b_umpire_name" TEXT,\n "3b_umpire_id" TEXT,\n "3b_umpire_name" TEXT,\n "lf_umpire_id" TEXT,\n "lf_umpire_name" TEXT,\n "rf_umpire_id" TEXT,\n "rf_umpire_name" TEXT,\n "v_manager_id" TEXT,\n "v_manager_name" TEXT,\n "h_manager_id" TEXT,\n "h_manager_name" TEXT,\n "winning_pitcher_id" TEXT,\n "winning_pitcher_name" TEXT,\n "losing_pitcher_id" TEXT,\n "losing_pitcher_name" TEXT,\n "saving_pitcher_id" TEXT,\n "saving_pitcher_name" TEXT,\n "winning_rbi_batter_id" TEXT,\n "winning_rbi_batter_id_name" TEXT,\n "v_starting_pitcher_id" TEXT,\n "v_starting_pitcher_name" TEXT,\n "h_starting_pitcher_id" TEXT,\n "h_starting_pitcher_name" TEXT,\n "v_player_1_id" TEXT,\n "v_player_1_name" TEXT,\n "v_player_1_def_pos" REAL,\n "v_player_2_id" TEXT,\n "v_player_2_name" TEXT,\n "v_player_2_def_pos" REAL,\n "v_player_3_id" TEXT,\n "v_player_3_name" TEXT,\n "v_player_3_def_pos" REAL,\n "v_player_4_id" TEXT,\n "v_player_4_name" TEXT,\n "v_player_4_def_pos" REAL,\n "v_player_5_id" TEXT,\n "v_player_5_name" TEXT,\n "v_player_5_def_pos" REAL,\n "v_player_6_id" TEXT,\n "v_player_6_name" TEXT,\n "v_player_6_def_pos" REAL,\n "v_player_7_id" TEXT,\n "v_player_7_name" TEXT,\n "v_player_7_def_pos" REAL,\n "v_player_8_id" TEXT,\n "v_player_8_name" TEXT,\n "v_player_8_def_pos" REAL,\n "v_player_9_id" TEXT,\n "v_player_9_name" TEXT,\n "v_player_9_def_pos" REAL,\n "h_player_1_id" TEXT,\n "h_player_1_name" TEXT,\n "h_player_1_def_pos" REAL,\n "h_player_2_id" TEXT,\n "h_player_2_name" TEXT,\n "h_player_2_def_pos" REAL,\n "h_player_3_id" TEXT,\n "h_player_3_name" TEXT,\n "h_player_3_def_pos" REAL,\n "h_player_4_id" TEXT,\n "h_player_4_name" TEXT,\n "h_player_4_def_pos" REAL,\n "h_player_5_id" TEXT,\n "h_player_5_name" TEXT,\n "h_player_5_def_pos" REAL,\n "h_player_6_id" TEXT,\n "h_player_6_name" TEXT,\n "h_player_6_def_pos" REAL,\n "h_player_7_id" TEXT,\n "h_player_7_name" TEXT,\n "h_player_7_def_pos" REAL,\n "h_player_8_id" TEXT,\n "h_player_8_name" TEXT,\n "h_player_8_def_pos" REAL,\n "h_player_9_id" TEXT,\n "h_player_9_name" TEXT,\n "h_player_9_def_pos" REAL,\n "additional_info" TEXT,\n "acquisition_info" TEXT\n, game_id TEXT)

In [45]:

```
#team_appearance table
create = '''
    CREATE TABLE IF NOT EXISTS team_appearance(
       CREATE TABLE IF NOT EXISTS team_appearance(

       team_id TEXT,
       game_id TEXT,
       home BOOLEAN,
       league_id TEXT,
       score INTEGER,
       line_score TEXT,
       at_bats INTEGER,
       hits INTEGER,
       doubles INTEGER,
       triples INTEGER,
       homeruns INTEGER,
       rbi INTEGER,
       sacrifice_hits INTEGER,
       sacrifice_flies INTEGER,
       hit_by_pitch INTEGER,
       walks INTEGER,
       intentional_walks INTEGER,
       strikeouts INTEGER,
       stolen_bases INTEGER,
       caught_stealing INTEGER,
       grounded_into_double INTEGER,
       first_catcher_interference INTEGER,
       left_on_base INTEGER,
       pitchers_used INTEGER,
       individual_earned_runs INTEGER,
       team_earned_runs INTEGER,
       wild_pitches INTEGER,
       balks INTEGER,
       putouts INTEGER,
       assists INTEGER,
       errors INTEGER,
       passed_balls INTEGER,
       double_plays INTEGER,
       triple_plays INTEGER,
       PRIMARY KEY (team_id, game_id),
       FOREIGN KEY (team_id) REFERENCES team(team_id),
       FOREIGN KEY (game_id) REFERENCES game(game_id),
       FOREIGN KEY (league_id) REFERENCES league(league_id)
       );
'''

run_command(create)

#Use a union to include columns for BOTH the home team and visiting team
#for the home teams, insert '1' for the BOOLEAN column and '0' for visitors
fill = '''
       INSERT OR IGNORE INTO team_appearance
          SELECT
            h_name,
            game_id,
            1 AS home,
            h_league,
            h_score,
            h_line_score,
            h_at_bats,
            h_hits,
            h_doubles,
            h_triples,
            h_homeruns,
            h_rbi,
            h_sacrifice_hits,
            h_sacrifice_flies,
            h_hit_by_pitch,
            h_walks,
            h_intentional_walks,
            h_strikeouts,
            h_stolen_bases,
            h_caught_stealing,
            h_grounded_into_double,
            h_first_catcher_interference,
            h_left_on_base,
            h_pitchers_used,
            h_individual_earned_runs,
            h_team_earned_runs,
            h_wild_pitches,
            h_balks,
            h_putouts,
            h_assists,
            h_errors,
            h_passed_balls,
            h_double_plays,
            h_triple_plays
          FROM game_log

       UNION

          SELECT
            v_name,
            game_id,
            0 AS home,
            v_league,
            v_score,
            v_line_score,
            v_at_bats,
            v_hits,
            v_doubles,
            v_triples,
            v_homeruns,
            v_rbi,
            v_sacrifice_hits,
            v_sacrifice_flies,
            v_hit_by_pitch,
            v_walks,
            v_intentional_walks,
            v_strikeouts,
            v_stolen_bases,
            v_caught_stealing,
            v_grounded_into_double,
            v_first_catcher_interference,
            v_left_on_base,
            v_pitchers_used,
            v_individual_earned_runs,
            v_team_earned_runs,
            v_wild_pitches,
            v_balks,
            v_putouts,
            v_assists,
            v_errors,
            v_passed_balls,
            v_double_plays,
            v_triple_plays
          FROM game_log;
'''
run_command(fill)
```

In [46]:

```
# ensure all games have been added, and that 'home' in each row only consists
of 1 or 0
run_query('select home, count(*) from team_appearance group by home')
```

Out[46]:

```
   home  count(*)
0     0    171907
1     1    171907
```

In [47]:

```
run_query('select * from team_appearance where team_id = "ALT"')
```

Out[47]:

	team_id	game_id	home	league_id	score	line_score	at_bats	hits	doubles
0	ALT	18840417CNU0	0	UA	2	None	None	None	None
1	ALT	18840418CNU0	0	UA	2	None	None	None	None
2	ALT	18840419CNU0	0	UA	6	None	None	None	None
3	ALT	18840424SLU0	0	UA	2	None	None	None	None
4	ALT	18840426SLU0	0	UA	3	None	None	None	None
5	ALT	18840427SLU0	0	UA	1	None	None	None	None
6	ALT	18840428SLU0	0	UA	1	None	None	None	None
7	ALT	18840430ALT0	1	UA	2	None	None	None	None
8	ALT	18840502ALT0	1	UA	3	None	None	None	None
9	ALT	18840503ALT0	1	UA	5	None	None	None	None
10	ALT	18840505ALT0	1	UA	2	None	None	None	None
11	ALT	18840510ALT0	1	UA	9	None	None	None	None
12	ALT	18840512ALT0	1	UA	3	None	None	None	None
13	ALT	18840514ALT0	1	UA	2	None	None	None	None
14	ALT	18840515ALT0	1	UA	7	None	None	None	None
15	ALT	18840516ALT0	1	UA	6	None	None	None	None
16	ALT	18840517ALT0	1	UA	8	None	None	None	None
17	ALT	18840521ALT0	1	UA	3	None	None	None	None
18	ALT	18840523ALT0	1	UA	8	None	None	None	None
19	ALT	18840524ALT0	1	UA	3	None	None	None	None
20	ALT	18840526ALT0	1	UA	6	None	None	None	None
21	ALT	18840527ALT0	1	UA	3	None	None	None	None
22	ALT	18840529ALT0	1	UA	0	None	None	None	None
23	ALT	18840530ALT0	1	UA	0	None	None	None	None
24	ALT	18840531ALT0	1	UA	3	None	None	None	None

In [48]:

```
# lets make sure that not all of the stats are 'None'
run_query('''
          SELECT *
          FROM team_appearance
          WHERE at_bats
          NOT NULL
          LIMIT 5''')
```

Out[48]:

	team_id	game_id	home	league_id	score	line_score	at_bats	hits	doubles	tri
0	ANA	20000403ANA0	1	AL	2	010000001	35	10	1	
1	ANA	20000404ANA0	1	AL	3	000003000	36	10	0	
2	ANA	20000405ANA0	1	AL	12	12610110x	33	12	4	
3	ANA	20000407ANA0	1	AL	7	30000310x	32	9	2	
4	ANA	20000408ANA0	1	AL	7	20000401x	35	13	4	

It may be the case that games from roughly 1800-1950 are more likely to contain missing data within the team_appearance table

The person_appearance table

Perhaps the most difficult table to fill. Now we have created all of the tables it depends on (due to foreign keys) we can attempt to fill it up. Much like the way we filled the team_appearance table we will need to make use of union. Except this time instead of performing a union on two large subsets of game_log, we will instead be performing a union on all of the positional columns in game_log (hp_umpire through to v_player_1 through to h_player_9).
On top of the umpire/manager details, a total of 36 inserts for each game will be necessary for the positions each player held.
There are 2 teams both consisting of 9 players, each of whom have 2 positions(offensive, defensive) 2x2x9 = 36

In [49]:

```
#create the person_appearance table

create = '''
         CREATE table IF NOT EXISTS person_appearance(
         appearance_id INTEGER PRIMARY KEY,
         person_id TEXT,
         team_id TEXT,
         game_id TEXT,
         role_id TEXT,
         FOREIGN KEY (person_id) REFERENCES person(person_id),
         FOREIGN KEY (team_id) REFERENCES team(team_id),
         FOREIGN KEY (game_id) REFERENCES game(game_id),
         FOREIGN KEY (role_id) REFERENCES role(role_id)
         );
 '''
run_command(create)
```

In [50]:

```
show_tables()
```

Out[50]:

	name	type
0	park_codes	table
1	team_codes	table
2	game_log	table
3	person_codes	table
4	person	table
5	park	table
6	league	table
7	role	table
8	team	table
9	game	table
10	team_appearance	table
11	person_appearance	table

In [51]:

```
run_query('select * from game_log limit 1')
```

Out[51]:

	date	number_of_game	day_of_week	v_name	v_league	v_game_number	h_name
0	18710504	0	Thu	CL1	None	1	FW1

In [52]:

```
#first of all fill up the umpires
fill_umpires = '''
     INSERT OR IGNORE INTO person_appearance(
         person_id,
         team_id,
         game_id,
         role_id
     )

         SELECT
           hp_umpire_id,
           NULL,
           game_id,
           "UHP"
         FROM game_log
         WHERE hp_umpire_id IS NOT NULL

      UNION

         SELECT
           [1b_umpire_id],
           NULL,
           game_id,
           "U1B"
         FROM game_log
         WHERE [1b_umpire_id] IS NOT NULL

      UNION

         SELECT
           [2b_umpire_id],
           NULL,
           game_id,
           "U2B"
         FROM game_log
         WHERE [2b_umpire_id] IS NOT NULL

      UNION

         SELECT
           [3b_umpire_id],
           NULL,
           game_id,
           "U3B"
         FROM game_log
         WHERE [3b_umpire_id] IS NOT NULL

      UNION

         SELECT
           lf_umpire_id,
           NULL,
           game_id,
           "ULF"
         FROM game_log
         WHERE lf_umpire_id IS NOT NULL

      UNION

         SELECT
           rf_umpire_id,
           NULL,
           game_id,
           "URF"
         FROM game_log
         WHERE rf_umpire_id IS NOT NULL

'''

run_command(fill_umpires)
```

In [53]:

```
#now for the managers

fill_managers = '''
        INSERT OR IGNORE INTO person_appearance(
          person_id,
          team_id,
          game_id,
          role_id
        )

          SELECT
            v_manager_id,
            v_name,
            game_id,
            "MM"
          FROM game_log
          WHERE v_manager_id IS NOT NULL

        UNION

          SELECT
            h_manager_id,
            h_name,
            game_id,
            "MM"
          FROM game_log
          WHERE h_manager_id IS NOT NULL

'''
run_command(fill_managers)
```

In [54]:

```
run_query('select * from game_log limit 1')
```

Out[54]:

	date	number_of_game	day_of_week	v_name	v_league	v_game_number	h_name
0	18710504	0	Thu	CL1	None	1	FW1

In [55]:

```
# insert 'awards' such as winning pitcher, losing pitcher etc
# create command
fill_awards = '''

         INSERT OR IGNORE INTO person_appearance(
         person_id,
         team_id,
         game_id,
         role_id
         )
         SELECT
           winning_pitcher_id,
           CASE
               WHEN h_score > v_score THEN h_name
               ELSE v_name
               END AS team_id,
           game_id,
           "AWP"
           FROM game_log
           WHERE winning_pitcher_id IS NOT NULL

        UNION

         SELECT
           losing_pitcher_id,
           CASE
               WHEN h_score < v_score THEN h_name
               ELSE v_name
               END AS team_id,
           game_id,
           "ALP"
           FROM game_log
           WHERE losing_pitcher_id IS NOT NULL

        UNION

         SELECT
           saving_pitcher_id,
           CASE
               WHEN h_score > v_score THEN h_name
               ELSE v_name
               END AS team_id,
           game_id,
           "ASP"
           FROM game_log
           WHERE saving_pitcher_id IS NOT NULL

        UNION

         SELECT
           winning_rbi_batter_id,
           CASE
               WHEN h_score > v_score THEN h_name
               ELSE v_name
               END AS team_id,
           game_id,
           "AWB"
           FROM game_log
           WHERE winning_rbi_batter_id IS NOT NULL

'''

# insert into table

run_command(fill_awards)
```

Outfield players

Considering there are 9 players for each team(and therefore 9 columns to insert), it does not make sense to insert their respective positions into the table via means of 18 UNION clauses (this will be unsightly). Instead we will make a generic template and use a for loop

In [56]:

```
#insert player positions

template = '''
        INSERT OR IGNORE INTO person_appearance(
          person_id,
          team_id,
          game_id,
          role_id
        )

          SELECT
              {hv}_player_{num}_id,
              {hv}_name,
              game_id,
              "O{num}"
           FROM game_log
           WHERE {hv}_player_{num}_id IS NOT NULL

         UNION

           SELECT
              {hv}_player_{num}_id,
              {hv}_name,
              game_id,
              "D" || CAST({hv}_player_{num}_def_pos AS INT)
           FROM game_log
           WHERE {hv}_player_{num}_id IS NOT NULL


'''

# loop through all combinations . h_1, ......v_9
for hv in ["h", "v"]:
    for i in range(1,10):
        mapping={"hv":hv,
                 "num":i}
        run_command(template.format(**mapping))
```

In [57]:

```
#check atleast one of each role has been added
run_query('select * from person_appearance group by role_id ')
```

Out[57]:

	appearance_id	person_id	team_id	game_id	role_id
0	1302389	zuveg101	DET	19550703KC10	ALP
1	1302378	zuveg101	DET	19540824BOS0	ASP
2	1302391	zuvep001	CLE	19890927SEA0	AWB
3	1302392	zycht001	SEA	20160421CLE0	AWP
4	6372483	zuveg101	DET	19540925CLE0	D1
5	6371697	zeilt001	NYA	20030802OAK0	D10
6	6372451	zunim001	SEA	20150827CHA0	D2
7	6371953	zinta001	HOU	20020902TEX0	D3
8	6372209	zobrb001	TBA	20080520OAK0	D4
9	6371695	zeilt001	NYA	20030709CLE0	D5
10	6372553	zuvep001	NYA	19860729MIL0	D6
11	6372229	zobrb001	TBA	20080907TOR0	D7
12	6370423	younc004	ARI	20090524OAK0	D8
13	6368379	wrigg001	TEX	19850519CHA0	D9
14	868217	zimmd101	TEX	19820728TEX0	MM
15	4119152	zuvep001	ATL	19851005SFN0	O1
16	4400828	zwild101	CHN	19160629SLN0	O2
17	4682504	zwild101	CHN	19160630SLN0	O3
18	4964180	zwild101	CHF	19150819BUF0	O4
19	5245856	zwild101	CHF	19140704IND2	O5
20	5527532	zwild101	CHA	19100910SLA2	O6
21	5809208	zwild101	CHA	19100902DET0	O7
22	6090884	zuvep001	NYA	19870718TEX0	O8
23	6372554	zuvep001	NYA	19860729MIL0	O9
24	524358	zimmc101	None	19040828SLN1	U1B
25	524125	younl901	None	20070721CHN0	U2B
26	524124	younl901	None	20070720CHN0	U3B
27	524403	zimmc101	None	19041009SLN2	UHP
28	510653	weyel901	None	19610929PIT0	ULF
29	521070	wolfj901	None	20071001COL0	URF

In [58]:

```
# show player/person lineup positions for the very last game in the database
run_query('''
       SELECT
       pa.*,
       role.name,
       role.category
       FROM person_appearance pa INNER JOIN
       role ON pa.role_id = role.role_id
       WHERE pa.game_id = (SELECT
                           MAX(game_id)
                           FROM
                           game)
       ORDER BY pa.team_id

''')
```

Out[58]:

	appearance_id	person_id	team_id	game_id	role_id	name	category
0	255326	kellj901	None	20161002WAS0	U3B	Third Base	umpire
1	350889	onorb901	None	20161002WAS0	U2B	Second Base	umpire
2	377885	porta901	None	20161002WAS0	U1B	First Base	umpire
3	480195	tumpj901	None	20161002WAS0	UHP	Home Plate	umpire
4	726702	mattd001	MIA	20161002WAS0	MM	Manager	manager
5	908963	brica001	MIA	20161002WAS0	ALP	Losing Pitcher	award
6	3943347	gordd002	MIA	20161002WAS0	D4	2nd Base	defense
7	3943348	gordd002	MIA	20161002WAS0	O1	Batter 1	offense
8	4365731	telit001	MIA	20161002WAS0	D2	Catcher	defense
9	4365732	telit001	MIA	20161002WAS0	O2	Batter 2	offense
10	4596503	pradm001	MIA	20161002WAS0	D5	3rd Base	defense
11	4596504	pradm001	MIA	20161002WAS0	O3	Batter 3	offense
12	4958975	yelic001	MIA	20161002WAS0	D8	Center Field	defense
13	4958976	yelic001	MIA	20161002WAS0	O4	Batter 4	offense
14	4987965	bourj002	MIA	20161002WAS0	D3	1st Base	defense
15	4987966	bourj002	MIA	20161002WAS0	O5	Batter 5	offense
16	5475809	scrux001	MIA	20161002WAS0	D7	Left Field	defense
17	5475810	scrux001	MIA	20161002WAS0	O6	Batter 6	offense
18	5644793	hoodd001	MIA	20161002WAS0	D9	Right Field	defense
19	5644794	hoodd001	MIA	20161002WAS0	O7	Batter 7	offense
20	5919043	hecha001	MIA	20161002WAS0	D6	Shortstop	defense
21	5919044	hecha001	MIA	20161002WAS0	O8	Batter 8	offense

22	6227913	koeht001	MIA	20161002WAS0	D1	Pitcher	defense
23	6227914	koeht001	MIA	20161002WAS0	O9	Batter 9	offense
24	537064	baked002	WAS	20161002WAS0	MM	Manager	manager
25	962790	difow001	WAS	20161002WAS0	AWB	Winning RBI Batter	award
26	1126269	melam001	WAS	20161002WAS0	ASP	Saving Pitcher	award
27	1217917	schem001	WAS	20161002WAS0	AWP	Winning Pitcher	award
28	1555003	turnt001	WAS	20161002WAS0	D8	Center Field	defense
29	1555004	turnt001	WAS	20161002WAS0	O1	Batter 1	offense
30	1796783	reveb001	WAS	20161002WAS0	D7	Left Field	defense
31	1796784	reveb001	WAS	20161002WAS0	O2	Batter 2	offense
32	1967453	harpb003	WAS	20161002WAS0	D9	Right Field	defense
33	1967454	harpb003	WAS	20161002WAS0	O3	Batter 3	offense
34	2428359	zimmr001	WAS	20161002WAS0	D3	1st Base	defense
35	2428360	zimmr001	WAS	20161002WAS0	O4	Batter 4	offense
36	2494127	drews001	WAS	20161002WAS0	D5	3rd Base	defense
37	2494128	drews001	WAS	20161002WAS0	O5	Batter 5	offense
38	2772389	difow001	WAS	20161002WAS0	D4	2nd Base	defense
39	2772390	difow001	WAS	20161002WAS0	O6	Batter 6	offense
40	3064661	espid001	WAS	20161002WAS0	D6	Shortstop	defense
41	3064662	espid001	WAS	20161002WAS0	O7	Batter 7	offense
42	3421073	lobaj001	WAS	20161002WAS0	D2	Catcher	defense
43	3421074	lobaj001	WAS	20161002WAS0	O8	Batter 8	offense
44	3785443	schem001	WAS	20161002WAS0	D1	Pitcher	defense
45	3785444	schem001	WAS	20161002WAS0	O9	Batter 9	offense

final cleanup

We've now created all normalized tables and inserted all of our data!

Our last task is to remove the tables we created to import the original CSVs.

In [59]:

```
show_tables()
```

Out[59]:

	name	type
0	park_codes	table
1	team_codes	table
2	game_log	table
3	person_codes	table
4	person	table
5	park	table
6	league	table
7	role	table
8	team	table
9	game	table
10	team_appearance	table
11	person_appearance	table

In [60]:

```
# drop our original tables, use our 'tables' dictionary from earlier, cell 22

for table in tables:
    run_command('DROP TABLE {}'.format(table))
```

In [61]:

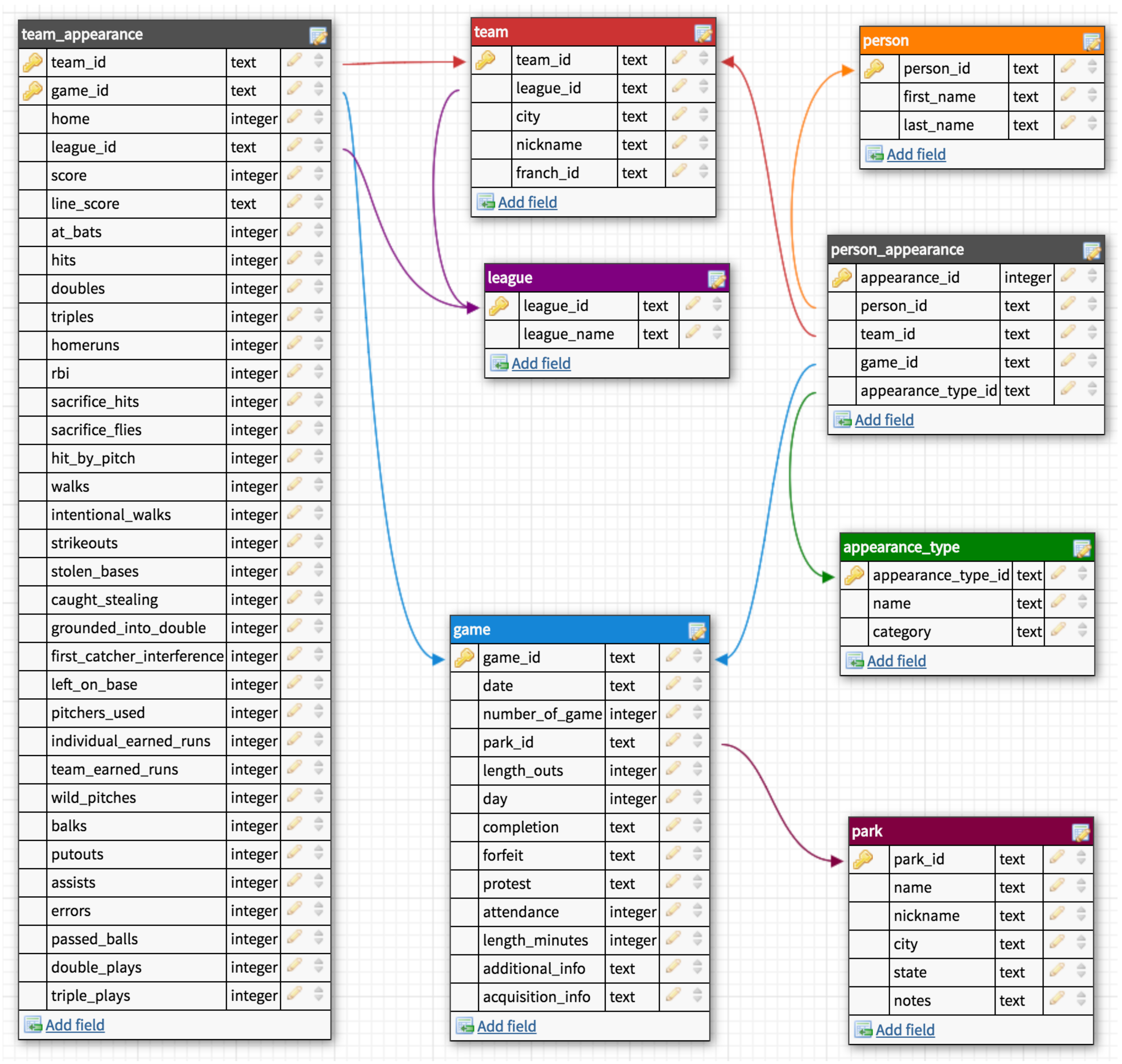
```
show_tables()
```

Out[61]:

	name	type
0	person	table
1	park	table
2	league	table
3	role	table
4	team	table
5	game	table
6	team_appearance	table
7	person_appearance	table

Little extra fun

lets see how we can use our database to find out simple statistics



team_appearance	
team_id	text
game_id	text
home	integer
league_id	text
score	integer
line_score	text
at_bats	integer
hits	integer
doubles	integer
triples	integer
homeruns	integer
rbi	integer
sacrifice_hits	integer
sacrifice_flies	integer
hit_by_pitch	integer
walks	integer
intentional_walks	integer
strikeouts	integer
stolen_bases	integer
caught_stealing	integer
grounded_into_double	integer
first_catcher_interference	integer
left_on_base	integer
pitchers_used	integer
individual_earned_runs	integer
team_earned_runs	integer
wild_pitches	integer
balks	integer
putouts	integer
assists	integer
errors	integer
passed_balls	integer
double_plays	integer
triple_plays	integer
Add field	

team	
team_id	text
league_id	text
city	text
nickname	text
franch_id	text
Add field	

league	
league_id	text
league_name	text
Add field	

game	
game_id	text
date	text
number_of_game	integer
park_id	text
length_outs	integer
day	integer
completion	text
forfeit	text
protest	text
attendance	integer
length_minutes	integer
additional_info	text
acquisition_info	text
Add field	

person	
person_id	text
first_name	text
last_name	text
Add field	

person_appearance	
appearance_id	integer
person_id	text
team_id	text
game_id	text
appearance_type_id	text
Add field	

appearance_type	
appearance_type_id	text
name	text
category	text
Add field	

park	
park_id	text
name	text
nickname	text
city	text
state	text
notes	text
Add field	

In [62]:

```
#lets see which team won the most game over all of our records

run_command('''
            CREATE VIEW game_winners AS

            SELECT
            CASE
                WHEN away_team.score > home_team.score THEN away_team.team_id
                WHEN away_team.score < home_team.team_id THEN home_team.score
                ELSE NULL
                END AS winner,
            away_team.game_id

            FROM

            (SELECT
            *
            FROM game g
            INNER JOIN team_appearance ta
            ON g.game_id = ta.game_id
            WHERE ta.home = 0) away_team

            INNER JOIN team_appearance home_team
            ON away_team.game_id = home_team.game_id
            WHERE home_team.home = 1


         ''')
```

In [63]:

```
# .....
run_query('''
            SELECT
            gw.winner,
            t.nickname,
            COUNT(gw.game_id) games_won

            FROM game_winners gw
            INNER JOIN team t
            ON gw.winner = t.team_id
            GROUP BY winner
            ORDER BY 3 DESC
            LIMIT 10

              ''')
```

Out[63]:

	winner	nickname	games_won
0	CHN	Cubs	4136
1	NYA	Yankees	3850
2	SLN	Cardinals	3803
3	PIT	Pirates	3780
4	PHI	Phillies	3651
5	CIN	Reds	3629
6	CLE	Indians	3514
7	DET	Tigers	3507
8	CHA	White Sox	3437
9	BOS	Red Sox	3420

The cubs have won the most games overall

END

My first attempt at creating a database with sql, took some time to get to grips with the normalization process but got there in the end.

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```