1. Summary and Business Case

This project looks to create a pitch usage database that tracks the types of pitches thrown by professional baseball players as well as the events surrounding their decisions to use certain pitches, while also monitoring the movement of these pitches, starting from the release point out of the pitchers hand up to the moment the pitch reaches the plate. The importance of creating such a database will let us better understand the physics behind each pitch thrown by major league pitchers, as well as understand the effects of pitch sequencing and game theory regarding individual pitcher-batter matchups and decisions made.

As such, we first need to track every game that has been played. Every game will have multiple events, but each event must be assigned to one game. Each event will involve two teams, a fielding team with a pitcher and a batting team with a batter involved. Only one batter and one pitcher can be involved in one single event, though there can be multiple separate events involving the same pitcher and batter. Because of this, each event must occur sequentially, tracking the inning, outs, plate appearance, pitch used, pitch of the sequence, result of the pitch (ball, foul, or strike), outcome (strike out, hit, walk, etc.), men on base (first, second, third), the score, and the plate appearance of the inning. This must be done for every pitch thrown in every single game.

Each pitch must be assigned to a pitch type and each pitch can only have one pitch type, but there can be multiple pitches with different pitch types. Each pitch must also be related to a single pitcher, though a pitcher can throw multiple pitches of different pitch types. Each pitch thrown must also be linked to one and only one event. However, the pitch must also be linked to the pitch of the next event, so we can analyze pitch sequencing. Each pitch within the first event must have a distinct release point, trajectory, and movement (both on the x and z plane). Likewise, each pitch will end in one result, but many results can be traced backed to each pitch thrown.

On the player side, each game will feature players from two teams. Only two teams may play in any single game, and at least nine players from each team must be present. However, if the home team is from the American League, at least ten players will have to appear (given the DH-rule of the American League.) We must also know, for each team, what position each player has on the field (using baseball’s position guide, 1 through 10.) Each position must be played by a player on each team, but, given pitching changes, pinch hitters, pinch runners, defensive replacements, and injuries, many players could end up playing a single position in any given game.

1. Conceptual Model

The database will need at least six tables that will work together to provide the necessary information for each plate appearance and every pitch that has been thrown throughout the seasons.

The Game table will provide the GameID, the date a game was played, and the Home and Away teams.

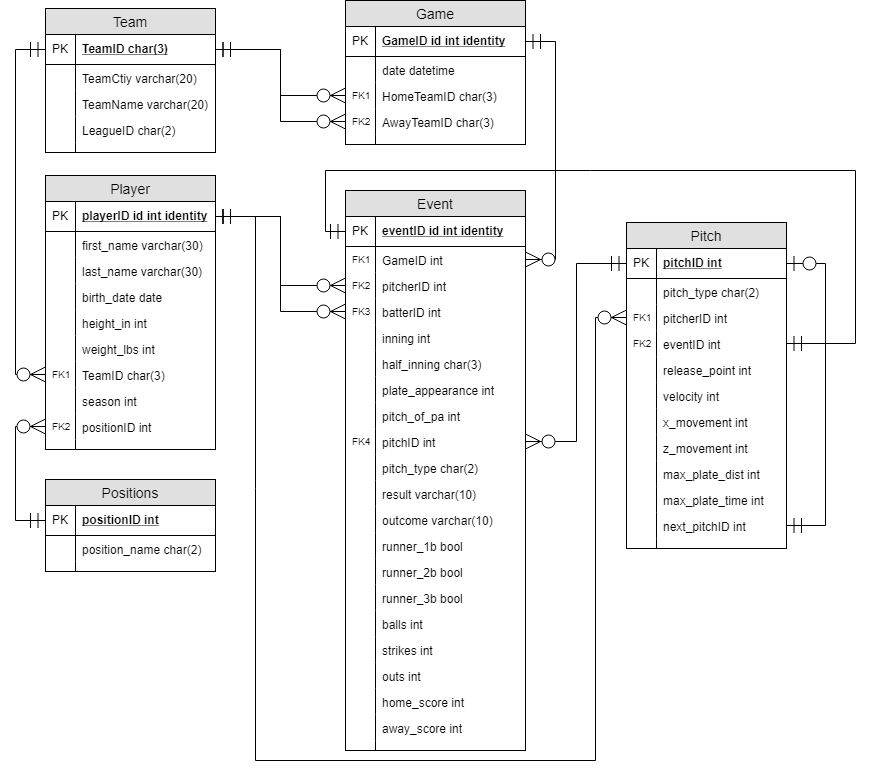
Both the HomeTeamID and the AwayTeamID will be fed from the Team table, which will have the TeamID, TeamCity, TeamName, and LeagueID, identifying the different teams that have played.

The GameID will work as the first foreign key that will help identify the different events that have occurred within a game. These events will be found in the Events table. Within this table, we will have the pitcherID, batterID, inning, half\_inning (top or bot), and plate\_appearance, provinding us with the context of every plate appearance that has occurred within the game. Similarly, the event will also have the pitchID of every pitch thrown in the plate appearance, along with the pitch\_type, the pitch\_of\_pa, the result of the pitch (whether it was a ball or a strike), and the outcome (which will be ball and strike for every pitch that doesn’t end the plate appearance; for the rest, it will show what happened). The events table will also show, through the use of boolean values, whether there were men on first, second, or third, the count (by adding the previous pitches thrown), the outs, and the scores for both home and away teams.

Both the pitcherID and batterID will be linked to the Player table, which will provide the information for each player: his first\_name, last\_name, birth\_date, height\_in (in inches), weight\_lbs (in pounds), TeamID (which will be linked to the Team table), season, and positionID for the main position the player played.

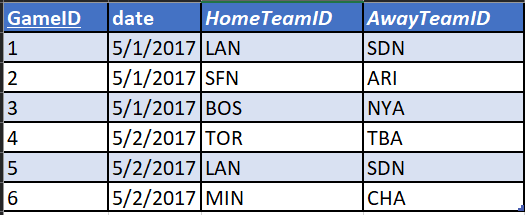
The positionID will be linked to the Position table where the names of each position will be taken from. PositionID will be an integer that goes from one to ten, using the standard notation for every baseball position (1 – pitcher, 2 – catcher, 3 – first base, 4 – second base, 5 – third base, 6 – shortstop, 7 – leftfield, 8 – centerfield, 9 – rightfield, 10 – designated hitter).

Going back to the events table, this table also features pitchID, which is a foreign key that will merge the Event table to the Pitch table. Within the Pitch table, we will find the pitchID and the pitch\_type of the pitch thrown. We will also find the pitcherID (which will link from the Player table) and the eventID (which will link from the Event table). Finally, we will have the information regarding the movement of the pitch: release\_point, velocity, x\_movement, z\_movement, max\_plate\_dist, max\_plate\_time, and next\_pitch (which will be the pitchID of the next pitch thrown in the sequence). This will help us identify pitch sequencing used by the different pitchers.



1. Normalization

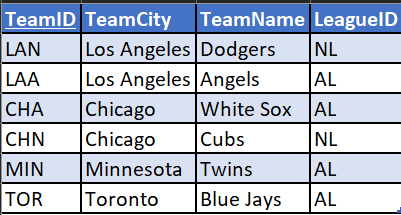
Starting with the Game table, we find that this is already in 3NF. We have the example data looking as follows:



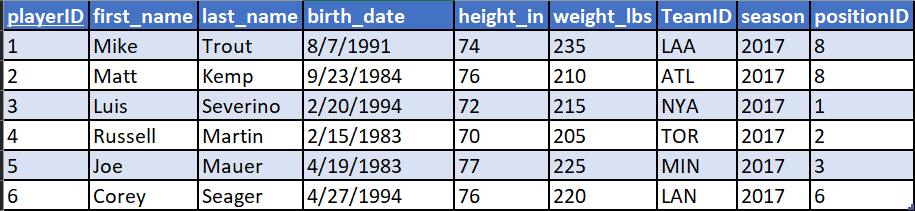
We end with the normalized form: Game (GameID, date, HomeTeamID, AwayTeamID)

Similarly, the Team, Player, and Position tables are also in 3NF:

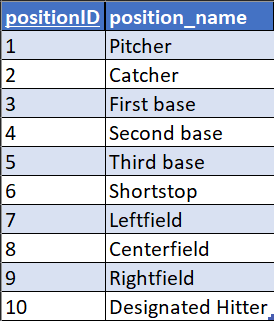
Team



Player



Position



These three tables are in the form:

Team (TeamID, TeamCity, TeamName, LeagueID)

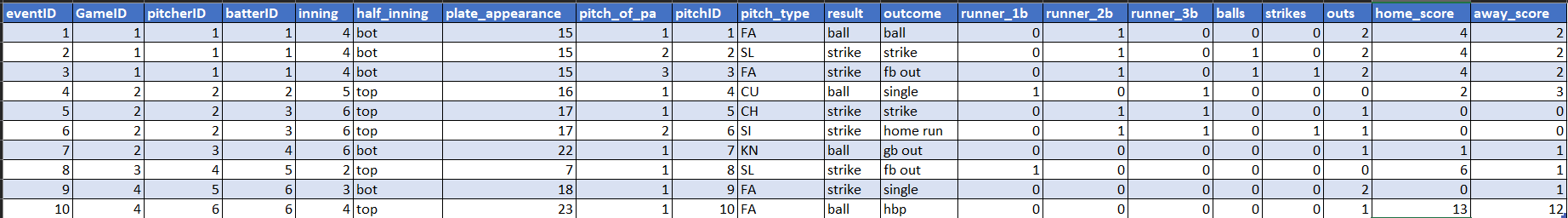
Player (playerID, first\_name, last\_name, birth\_date, height\_in, weight\_lbs, *TeamID*, season, *positionID*)

Position (positionID, position\_name)

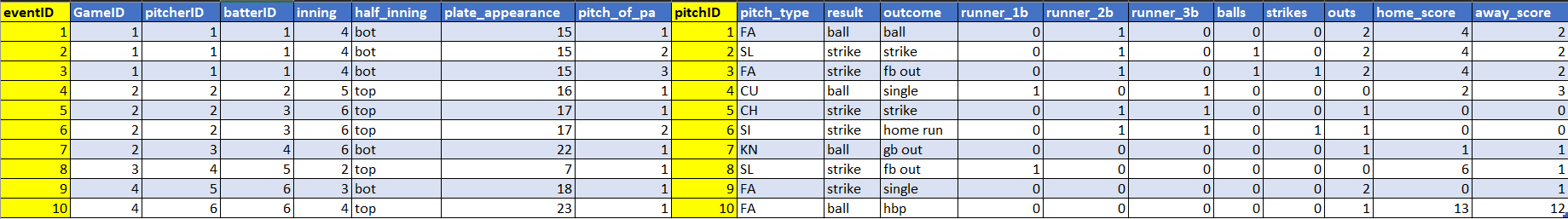
\*\*\*

The Event and Pitch tables, however, are not in 3NF, so we must convert them. Starting with the Event table, it currently sits like this:

Event (eventID, *GameID*, *pitcherID*, *batterID*, inning, half\_inning, plate\_appearance, pitch\_of\_pa, *pitchID*, pitch\_type, result, outcome, runner\_1b, runner\_2b, runner\_3b, balls, strikes, outs, home\_score, away\_score)

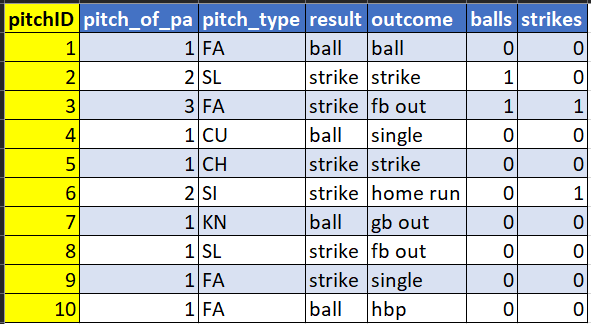


Looking over the table, the two principal dependencies that I can find are both the eventID column and the pitchID column.



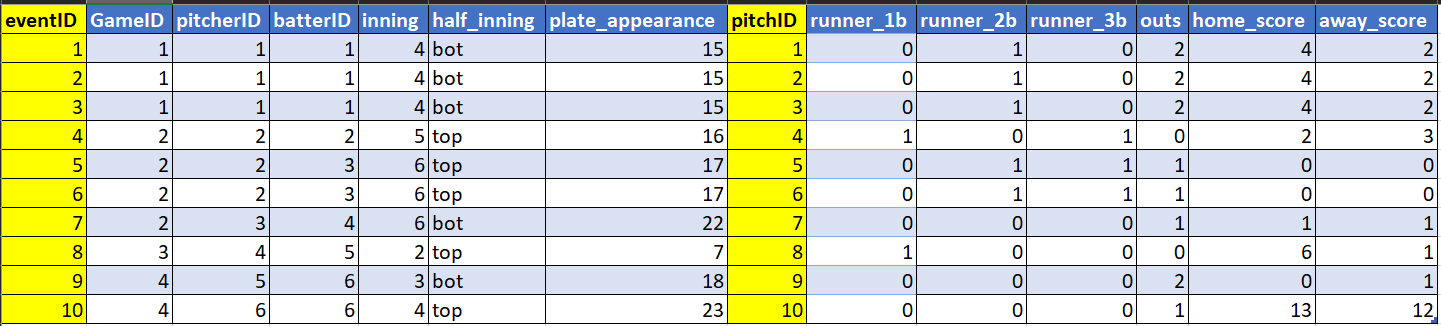
So, I’ll create a new table Pitch\_event to house the data for every pitch event:

Pitch\_event (pitchID, pitch\_of\_pa, pitch\_type, result, outcome, balls, strikes)



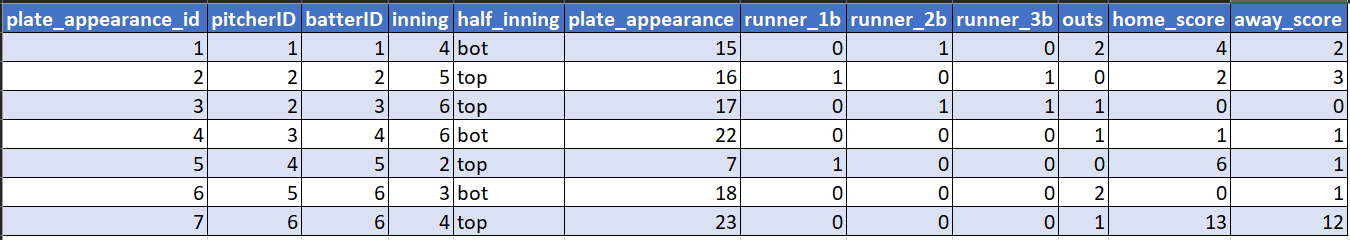
Meanwhile, the Event table looks something like this:

Event (eventID, *GameID*, *pitcherID*, *batterID*, inning, half\_inning, plate\_appearance, *pitchID*, runner\_1b, runner\_2b, runner\_3b, outs, home\_score, away\_score)

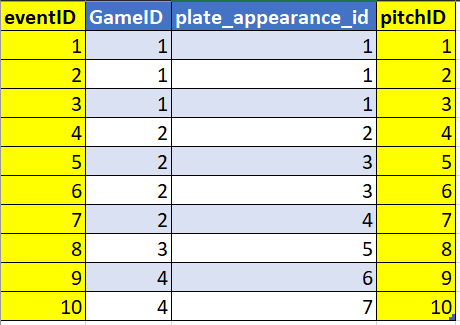


Looking at the table, it still feels that it is not completely in 3NF. So, I’ve decided to create a new table called Plate\_Appearance that gives the basic details of the plate appearance:

Plate\_Appearance (plate\_appearance\_id, *pitcherID*, *batterID*, inning, half\_inning, plate\_appearance, runner\_1b, runner\_2b, runner\_3b, outs, home\_score, away\_score)

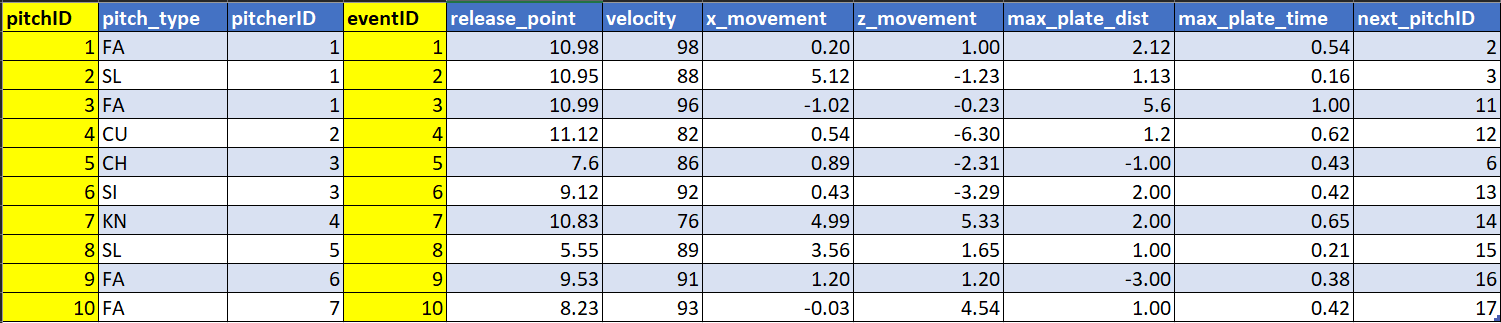


And, finally, the Event table looks like this: Event (eventID, *GameID*, *plate\_appearance\_id*, *pitchID*)

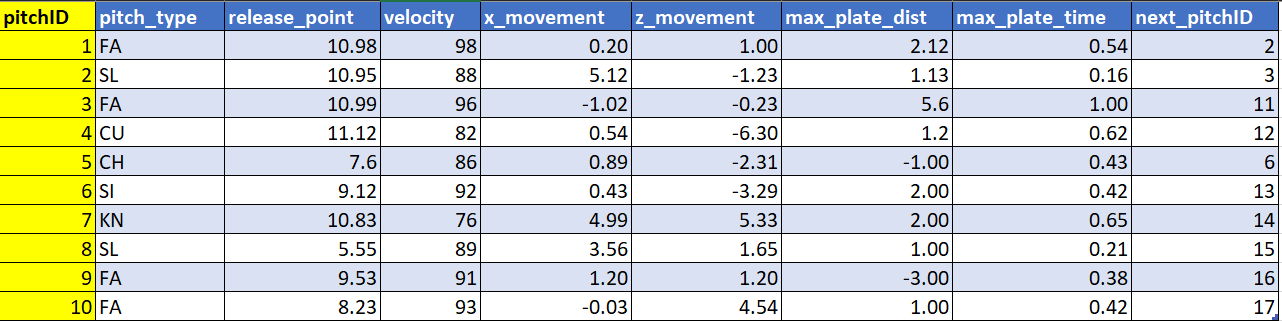


\*\*\*

The last table that needs to be converted to 3NF is the Pitch table. Currently, it looks like this:



Because we’re focusing on pitch tracking data here, there is really no other way to identify the pitch data by any other way that isn’t related to the pitchID, the eventID, or the pitcherID. Since the eventID is already being tracked in the Event table, which is connected to the Pitch table via pitchID, then we could remove this column. Similarly, the pitcherID is also present in the Plate\_Appearance table, which can be linked to the Pitch table via the Event table. So, we could also get rid of the pitcherID column, leaving us with the following table and 3NF:



Pitch (pitchID, pitch\_type, release\_point, velocity, x\_movement, z\_movement, max\_plate\_dist, max\_plate\_time, next\_pitchID)

The final diagram would look like this:

