MLB-DB

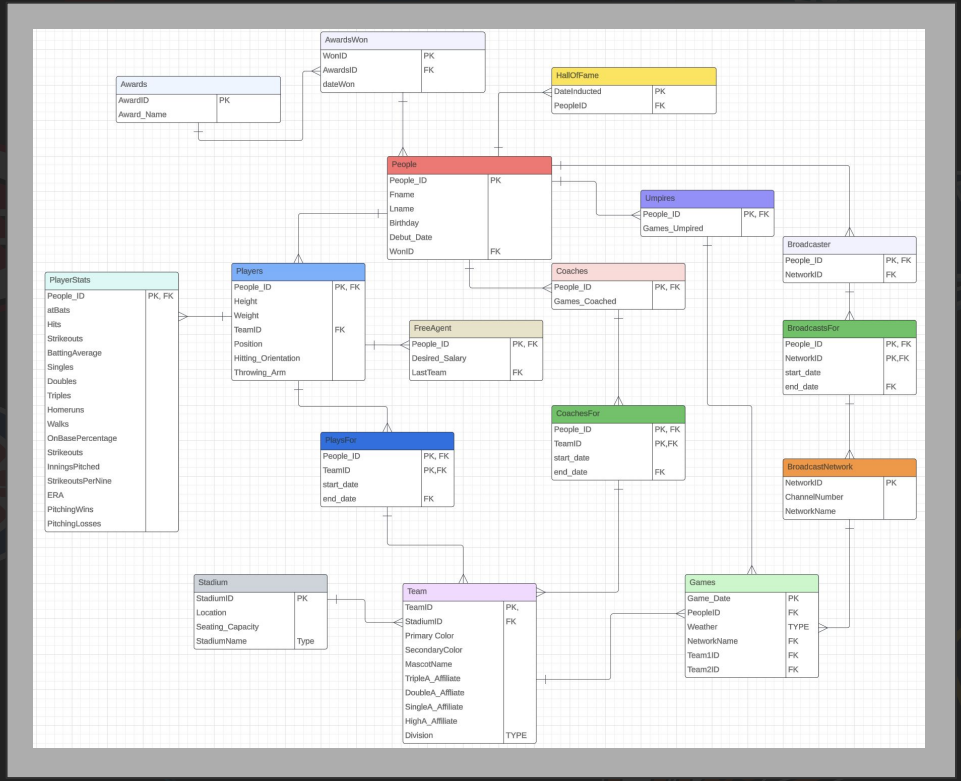
by Ethan Ondreicka

Table of Contents

Executive Summary

Major League Baseball hired me as a database architect to develop a new state-of-the-art database to keep track of all things related to MLB. This database was needed because of the league's latest expansion endeavors with the addition of 4 brand new teams, as well as their minor league affiliates. It is my job to make sure that I can account for all information a team may need, as well as any data a fan might want to see as well.

Entity-Relationship Diagram



Types

Division Type - The division type contains text entries of all the divisions a team may belong to

```
CREATE TYPE division AS ENUM ('NL East', 'NL Central', 'NL West', 'AL East', ‘AL Central’, ‘AL West’);
```

Orientation Type - the orientation type contains text entries of preferences when it comes to throwing and hitting

```
CREATE TYPE orientation AS ENUM ('lefty', 'righty', 'ambidextrous’);
```

Weather Type - the weather type contains all the different weather conditions that can happen during a game

```
CREATE TYPE orientation AS ENUM ('rainy', 'cloudy', 'sunny’, ‘snowy’, ‘windy’);
```

Tables

People Table - The people table contains all basic information about every single person in the database. The attributes are shared in the players, coaches, and umpires tables.

```
CREATE TABLE People (
   PeopleID    int not null,
   fName       text,
   lName       text,
   birthday    date,
   debutDate   date,
   WonID       int references AwardsWon(WonID),
 primary key(PeopleID)
);
```

Functional Dependencies:
PeopleID -> Fname, Lname, birthday, debutDate

	peopleid [PK] integer	fname text	lname text	birthday date	debutdate date	wonid integer
1	1	John	Lobster	1995-07-22	2018-04-12	[null]
2	2	Richard	Octo	1992-01-12	2012-08-18	[null]
3	3	Jim	Krill	2000-12-25	2022-09-01	[null]
4	4	Drew	Shark	1997-06-20	2016-08-30	5
5	5	Alan	Labouseur	1985-02-14	2006-03-30	1
6	6	Paul	Boats	1981-03-17	2002-06-22	[null]
7	7	Michael	Coral	1989-10-10	2002-04-12	[null]
8	8	Josh	Fish	2003-02-17	2024-03-29	4
9	9	Randy	Flounder	1995-07-26	2017-10-01	[null]
10	10	Carlos	Hook	1988-01-19	2012-06-05	[null]
11	11	Tim	Sea	2001-03-13	2023-07-12	[null]
12	12	Frank	Kelp	1973-11-24	1998-05-23	2
13	13	Kyle	Dolphin	1983-12-02	2006-08-17	[null]
14	14	Dominick	Osyter	1968-05-01	1993-02-12	3
15	15	Kevin	Shell	1990-08-12	2012-06-18	[null]

Tables

Players Table - The players table contains information about all the players in the database.

```
CREATE TABLE Players (
    PeopleID              int not null
                          references People(PeopleID),
    height_cm             decimal(8,3) not null,
    weight_kg             decimal(7,3) not null,
    position              text not null,
    hittingOrientation    orientation not null,
    throwingArm           orientation not null,
  primary key(PeopleID)
);
```

	peopleid [PK] integer	height_cm numeric (8,3)	weight_kg numeric (7,3)	position text	hittingorientation orientation	throwingarm orientation
1	1	190.500	80.000	Second Ba...	lefty	righty
2	2	165.100	65.317	Center Field	righty	righty
3	4	183.439	95.467	Shortstop	righty	ambidextrous
4	5	808.808	808.808	Pitcher	righty	righty
5	8	199.691	101.803	Pitcher	lefty	lefty
6	9	120.700	72.991	Right Field	righty	lefty
7	11	166.388	59.140	Catcher	righty	righty

Functional Dependencies:
PeopleID -> height_cm, weight_kg, position, hitting Orientation, throwingArm

Tables

Coaches Table - The coaches table contains information about all coaches in the database

```
CREATE TABLE Coaches (
    PeopleID        int not null
                    references People(PeopleID),
    gamesCoached    int,
  primary key(PeopleID)
);
```

	peopleid [PK] integer	gamescoached integer
1	6	1378
2	10	322
3	14	2001

Functional Dependencies:
PeopleID -> gamesCoached

Tables

Umpires Table - The umpires table contains information about all umpires in the database.

```
CREATE TABLE Umpires (
    PeopleID        int not null
                    references People(PeopleID),
    UmpireID        int not null,
    gamesUmpired    int,
  primary key(UmpireID)
);
```

	peopleid integer	umpireid [PK] integer	gamesumpired integer
1	7	1	287
2	15	2	677

Functional Dependencies:
PeopleID -> gamesUmpired

Tables

Broadcaster Table - The Broadcaster table contains information about all broadcasters in the database.

```
CREATE TABLE Broadcasters (
    PeopleID        int not null
                     references People(PeopleID),
    gamesBroadcasted int,
  primary key(PeopleID)
);
```

	peopleid [PK] integer	gamesbroadcasted integer
1	3	183
2	12	2001
3	13	971

Functional Dependencies:
PeopleID -> gamesBroadcasted

Tables

Free Agent Table - The Free Agent table contains all relevant information about all the free agents in the database.

```
CREATE TABLE FreeAgents (
    PeopleID        int not null
                    references People(PeopleID),
    desiredSalary   int,
    lastTeamID      int not null
                    references Team(TeamID),
  primary key(PeopleID)
);
```

	peopleid [PK] integer	desiredsalary integer	lastteamid integer
1	8	2500000	3
2	1	1375000	1

Functional Dependencies:
PeopleID -> desiredSalary, lastTeamID

Tables

Team Table - The team table contains all basic information about every single team in the database.

```
CREATE TABLE Team (
    TeamID              int not null,
    StadiumID           int not null
                        references Stadiums(StadiumID),
    teamName            text not null,
    primaryColor        text not null,
    secondaryColor      text not null,
    mascotName          text,
    tripleA_Affiliate   text not null,
    doubleA_Affiliate   text not null,
    singleA_Affiliate   text not null,
    highA_Affiliate     text not null,
    division            division not null,
  primary key(TeamID)
);
```

Functional Dependencies:
TeamID -> StadiumID, teamName, primaryColor, secondaryColor, mascotName, tripleA_Affiliate, doubleA_Affiliate, singleA_Affiliate, highA_Affiliate, division

	teamid [PK] integer	stadiumid integer	teamname text	primarycolor text	secondarycolor text	mascotname text	triplea_affiliate text	doublea_affiliate text	singlea_affiliate text	higha_affiliate text	division division
1	1	1	Liberty	Green	White	Lady Liberty	Vermont Trees	Alaska Yetis	Buffalo Rivers	Miami Sunsets	NL East
2	2	3	Retirees	Light Blue	Navy Blue	Grandpa Joe	Texas Chainsaws	Arizona Nothingness	Utah Mormons	Maine Mutants	AL East
3	3	2	Climbers	Olive Green	Brown	Rocky the Rock	Seattle Squares	Denver Dawgs	Kansas Nados	North Carolina Borings	NL West
4	4	4	Water Buffalos	Yellow	Black	Buddy the Buffalo	Minnesota Hogs	Georgia Peaches	Iowa Goats	Oregon Petals	AL Central

Tables

Stadium Table - The stadium table contains all basic information about every stadium in the database.

```
CREATE TABLE Stadiums (
    StadiumID           int not null,
    location            text not null,
    seatingCapacity     int not null,
    stadiumName         text not null,
  primary key(StadiumID)
);
```

	stadiumid [PK] integer	location text	seatingcapacity integer	stadiumname text
1	1	New York	53000	Liberty Park
2	2	Colorado	49000	Mountain Field
3	3	Florida	55000	Blue Ocean Stadium
4	4	Wyoming	38000	The Buffalo Dome

Functional Dependencies:
StadiumID -> location, seatingCapacity, stadiumName

Tables

Games Table - The games table contains all basic information about every game the Wyoming Buffalo play

```
CREATE TABLE Game (
    GameDate            date not null,
    weather             weather not null,
    NetworkID           int not null
                        references BroadcastNetworks(NetworkID),
    Team1ID             int not null references Team(TeamID),
    Team2ID             int not null references Team(TeamID),
    UmpireID            int not null references Umpires(UmpireID),
  primary key(GameDate)
);
```

	gamedate [PK] date	weather weather	networkid integer	team1id integer	team2id integer	umpireid integer
1	2024-05-12	sunny	3	1	4	1
2	2024-05-13	rainy	2	2	4	2
3	2024-05-14	snowy	2	1	4	1
4	2024-05-15	sunny	1	3	4	2
5	2024-05-17	cloudy	3	4	1	1
6	2024-05-18	sunny	1	4	3	2

Functional Dependencies:
GameDate -> weather, NetworkID, Team1ID, Team2ID, UmpireID

Tables

BroadcastNetwork Table - The BroadcastNetwork table contains all of the broadcast networks in the database.

```
CREATE TABLE BroadcastNetworks(
    NetworkID       int not null,
    channelNumber   int not null,
    networkName     text not null,
  primary key(NetworkID)
);
```

	networkid [PK] integer	channelnumber integer	networkname text
1	1	225	Sports Central
2	2	585	Bingus Sportus
3	3	75	Big Baseball Broadcasts

Functional Dependencies:
NetworkID -> channelNumber, networkName

Tables

Awards Table - The awards table contains all the awards a player, coach, or broadcaster may win during their careers.

```
CREATE TABLE Awards(
    AwardID          int not null,
    Award_Name       text not null,
  primary key(AwardID)
);
```

	awardid [PK] integer	award_name text
1	1	Silver Slugger
2	2	Cy Young
3	3	MVP
4	4	Coach of the Year
5	5	Broadcaster of the Year

Functional Dependencies: AwardID -> Award_Name

Tables

HallOfFame Table - The HallOfFame table contains any person who has been elected into the hall of fame. Players, Coaches, and Broadcasters all share one Hall of Fame.

```
CREATE TABLE HallOfFame(
    HallID          int not null,
    dateInducted    date not null,
    PeopleID        int not null
                    references People(PeopleID),
  primary key(HallID, PeopleID)
);
```

	hallid [PK] integer	dateinducted date	peopleid [PK] integer
1	1	2016-04-20	5
2	2	2000-08-17	14
3	3	2008-08-19	12

Functional Dependencies:
HallID, PeopleID -> dateInducted

Tables

PlayerStats Table - The PlayerStats table contains information and statistics about every player in the database.

```
CREATE TABLE PlayerStats (
    PeopleID        int not null references People(PeopleID),
    atBats          int,
    hits            int,
    strikeouts      int,
    battingAverage  decimal(5,3),
    singles         int,
    doubles         int,
    triples         int,
    homeruns        int,
    walks           int,
    onBasePercentage decimal (5,3),
    pitchingStrikeouts   int,
    inningsPitched       int,
    strikeoutsPerNine    decimal(5,2),
    earnedRuns           int,
    earnedRunAverage     decimal(5,2),
    pitchingWins         int,
    pitchingLosses       int,
  primary key(PeopleID)
);
```

Functional Dependencies:
PeopleID -> atBats, hits, strikeouts, battingAverage, singles, doubles, triples, homeruns, walks, onBasePercentage, pitchingStrikeouts, inningsPitched, strikeoutPerNine, earnedRuns, earnedRunAverage, pitchingWins, pitchingLosses

	peopleid [PK] integer	atbats integer	hits integer	strikeouts integer	battingaverage numeric (5,3)	singles integer	doubles integer	triples integer	homeruns integer	walks integer	onbasepercentage numeric (5,3)	pitchingstrikeouts integer	inningspitched integer	strikeoutspernine numeric (5,2)	earnedruns integer	earnedrunaverage numeric (5,2)	pitchingwins integer	pitchinglosses integer
1	1	891	212	189	[null]	184	20	3	5	102	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]
2	2	2082	550	289	[null]	300	74	42	134	200	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]
3	4	999	333	44	[null]	222	56	18	37	66	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]
4	5	1000	1000	0	[null]	0	0	0	1000	0	[null]	2700	900	[null]	0	[null]	100	0
5	8	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	72	62	[null]	13	[null]	7	2
6	9	3038	891	388	[null]	588	120	51	132	204	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]
7	11	512	220	61	[null]	78	40	9	93	36	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]

Tables

AwardsWon Table - The AwardsWon table contains all basic information about every single person who won an award

```
CREATE TABLE AwardsWon (
      WonID int not null,
    AwardsID int not null references Awards(AwardID),
      dateWon date not null,
  primary key(WonID)
);
```

	wonid [PK] integer	awardsid integer	datewon date
1	1	3	2010-10-30
2	2	5	2003-10-18
3	3	4	1999-10-02
4	4	2	2024-01-01
5	5	1	2019-10-03

Functional Dependencies:
WonID -> AwardsID, dateWon

Tables

PlaysFor Table - The PlaysFor junction table contains what team a player plays for.

```
CREATE TABLE PlaysFor (
    PeopleID        int not null references People(PeopleID),
    TeamID          int not null references Team(TeamID),
    start_date      date not null,
    end_date        date,
  primary key(PeopleID, TeamID)
);
```

	peopleid [PK] integer	teamid [PK] integer	start_date date	end_date date
1	1	1	2019-05-22	[null]
2	2	3	2016-08-12	[null]
3	4	3	2016-08-30	[null]
4	5	4	2020-06-17	[null]
5	8	4	2024-03-29	[null]
6	9	2	2017-10-01	2023-09-15
7	11	1	2023-07-12	[null]

Functional Dependencies:
PeopleID, TeamID -> start_date, end_date

Tables

CoachesFor Table - The CoachesFor junction table contains what team a coach coaches for.

```
CREATE TABLE CoachesFor (
    PeopleID        int not null references People(PeopleID),
    TeamID          int not null references Team(TeamID),
    start_date      date not null,
    end_date        date,
  primary key(PeopleID, TeamID)
);
```

	peopleid [PK] integer	teamid [PK] integer	start_date date	end_date date
1	6	2	2008-07-19	2022-09-18
2	10	1	2014-04-01	[null]
3	14	4	2000-05-22	[null]

Functional Dependencies:
PeopleID, TeamID -> start_date, end_date

Tables

BroadcastsFor Table - The BroadcastsFor table contains what network a Broadcaster broadcasts for.

```
CREATE TABLE BroadcastsFor (
    PeopleID        int not null references People(PeopleID),
    NetworkID       int not null
                    references BroadcastNetworks(NetworkID),
    start_date      date not null,
    end_date        date,
  primary key(PeopleID, NetworkID)
);
```

	peopleid [PK] integer	networkid [PK] integer	start_date date	end_date date
1	3	3	2023-01-19	[null]
2	12	1	2000-10-02	[null]
3	13	2	2015-03-27	2021-09-12

Functional Dependencies:
PeopleID, NetworkID -> start_date, end_date

Views

Team4Players - Shows all the players first names and last names, as well as the stadium name for the team whose TeamID is 4

```
CREATE VIEW Team4Players AS
SELECT p.fname, p.lname, s.stadiumName
FROM PEOPLE p
JOIN PlaysFor pf ON p.PeopleID = pf.PeopleID
JOIN Team t ON pf.TeamID = t.TeamID
JOIN Stadiums s ON t.stadiumID = s.StadiumID
WHERE t.TeamID = 4;
```

	fname text	lname text	stadiumname text
1	Alan	Labouseur	The Buffalo Dome
2	Josh	Fish	The Buffalo Dome

SunnyGamesForTeam4 - Shows all the games TeamID 4 has played where the weather was sunny

```
CREATE VIEW SunnyGameForTeam4 AS
SELECT *
FROM GAME
WHERE weather = 'sunny' AND Team2ID = 4;
```

	gamedate date	weather weather	networkid integer	team1id integer	team2id integer	umpireid integer
1	2024-05-12	sunny	3	1	4	1
2	2024-05-15	sunny	1	3	4	2

Reports

Report showing all individuals inducted into the Hall of Fame

```
SELECT p.fName, p.lName, h.dateInducted
FROM People p
JOIN HallOfFame h ON p.PeopleID = h.PeopleID
```

	fname text	lname text	dateinducted date
1	Alan	Labouseur	2016-04-20
2	Dominick	Osyter	2000-08-17
3	Frank	Kelp	2008-08-19

Report showing all Players on TeamID 3 and their positions

```
SELECT p.fName, p.lName, ply.position
FROM People p
JOIN PlaysFor pf ON p.PeopleID = pf.PeopleID
JOIN Players ply on p.PeopleID = ply.PeopleID
WHERE pf.TeamID = 3;
```

	fname text	lname text	position text
1	Richard	Octo	Center Field
2	Drew	Shark	Shortstop

Stored Procedures

`CalculateBattingAverage` - Self Explanatory, but uses the players batting stats to determine their batting average

```
CREATE OR REPLACE FUNCTION CalculateBattingAverage(player_id INT) RETURNS DECIMAL AS $$
DECLARE
    total_hits INT;
    total_at_bats INT;
    batting_avg DECIMAL(5, 3);
BEGIN
    SELECT SUM(hits), SUM(atBats) INTO total_hits, total_at_bats
    FROM PlayerStats
    WHERE PeopleID = player_id;

    IF total_at_bats > 0 THEN
        batting_avg := ROUND(total_hits::DECIMAL / total_at_bats, 3);
    ELSE
        batting_avg := 0.0;
    END IF;

    RETURN batting_avg;
END;
$$ LANGUAGE plpgsql;


SELECT CalculateBattingAverage(2)
SELECT CalculateBattingAverage(5)
```

	calculatebattingaverage numeric
1	0.264

	calculatebattingaverage numeric
1	1.000

Stored Procedures

`GetTeamRoster` - Gets the roster of an entire team, giving the first name, last name, and position of each player.

```
CREATE OR REPLACE FUNCTION GetTeamRoster(team_id INT) RETURNS TABLE (
    PlayerID INT,
    FirstName TEXT,
    LastName TEXT,
    PlayerPosition TEXT
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        p.PeopleID AS PlayerID,
        p.fName AS FirstName,
        p.lName AS LastName,
        ply.position AS PlayerPosition
    FROM
        People p
    JOIN
        PlaysFor pf ON p.PeopleID = pf.PeopleID
    JOIN
        Players ply ON p.PeopleID = ply.PeopleID
    WHERE
        pf.TeamID = team_id;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM GetTeamRoster(1);
SELECT * FROM GetTeamRoster(4);
```

	playerid integer	firstname text	lastname text	playerposition text
1	1	John	Lobster	Second Base
2	11	Tim	Sea	Catcher

	playerid integer	firstname text	lastname text	playerposition text
1	5	Alan	Labouseur	Pitcher
2	8	Josh	Fish	Pitcher

Triggers

`update_player_debut_date` - this trigger will automatically update the debut date of a person when they are added to the playsfor table

```
CREATE OR REPLACE FUNCTION update_player_debut_date()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE People
    SET debutDate = NEW.start_date
    WHERE People.PeopleID = NEW.PeopleID;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_debut_trigger
AFTER INSERT ON PlaysFor
FOR EACH ROW
EXECUTE FUNCTION update_player_debut_date();
```

Triggers

update_broadcasters_count - this trigger will automatically update the total number of games broadcasted by a broadcaster in the broadcaster table when a new entry is made in the BroadcasterFor table

```
CREATE OR REPLACE FUNCTION update_broadcasters_count()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE Broadcasters
    SET gamesBroadcasted = gamesBroadcasted + 1
    WHERE PeopleID = NEW.PeopleID;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_broadcasters_trigger
AFTER INSERT ON BroadcastsFor
FOR EACH ROW
EXECUTE FUNCTION update_broadcasters_count();
```

Security / Roles

Admin - The admin can view all the information on the database for all teams in Major League Baseball

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

Coach - The coach can view all the stats of the players who play for their team

```
CREATE ROLE coach;
GRANT SELECT, INSERT, UPDATE ON PlayerStats, PlaysFor
TO coach;
```

General Manager - The General Manager can view everything about their team including player stats, team info, stadium info, coach info as well as see Free Agents.

```
CREATE ROLE generalManager;
GRANT SELECT, INSERT, UPDATE ON PlayerStats, PlaysFor, Team, Stadium, FreeAgents
TO generalManager;
```

Implementation Notes

This database is an example of one that may be used for Major League Baseball. I wasn't able to add too many people for each of the different roles (player, umpire, coach, broadcaster), or else I would be here all day. For ease of use and example sake, I made sure to have at least a few people in each role with different attributes to make sure the data was unique to catch any discrepancies. This database can be expanded to include more teams, stadiums, and people very easily.

Known Problems

Currently, the only problems I can really think of is not having enough data inserted into the database, as well as the `PlayerStats` table being too big. I think it would be better if the `PlayerStats` table was broken down into two for hitting stats and pitching stats

Future Enhancements

In the future, I would probably add more people, teams, and stadiums to give the database more life. It feels like there are way too many tables for the amount of people I put in. I would also probably make a few more tables for the teams like an ‘injured list’ table. I could also some more attributes for players like salary and contract length which would be good for the more economical side of baseball.


Mets
Cardinals
REDS
RAYS
HOUSTON
ASTROS
UBS
Yankees
WASHINGTON
NATIONALS
Miami
Orioles