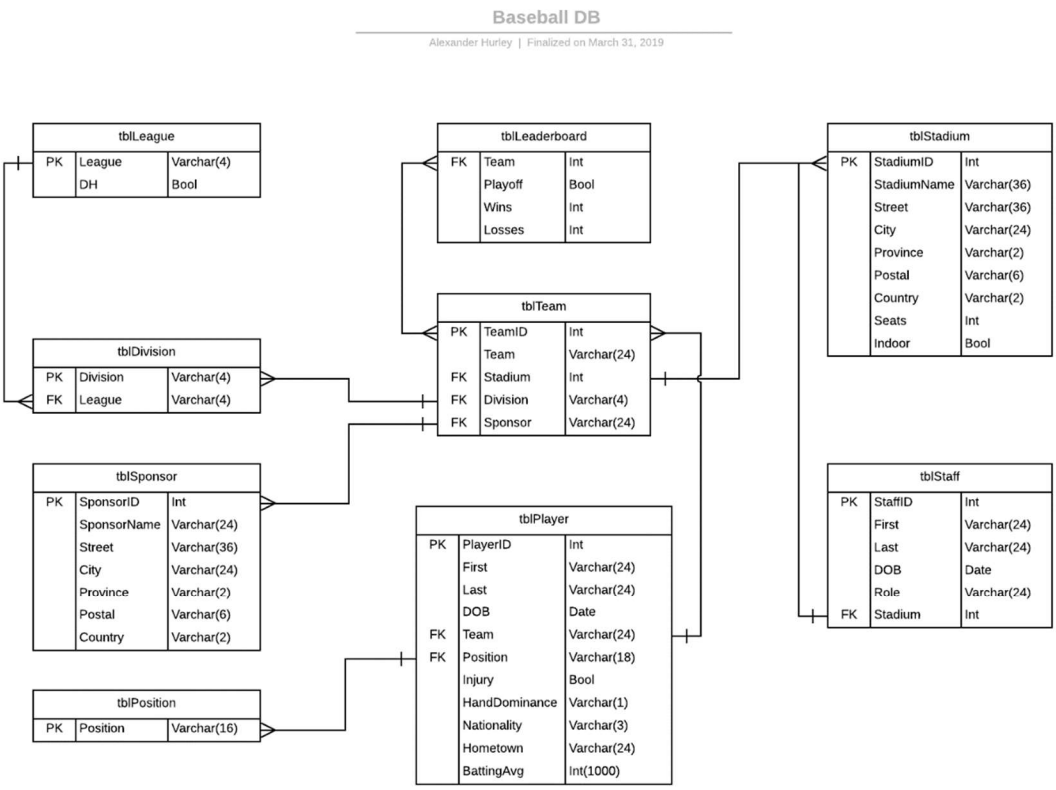
Database Systems Project
March 31st, 2019

Alexander Hurley

Part 1: ER Diagram and Overview



For this database, it was based on a small international baseball League. It is broken down into 2 Leagues, East and West respectively. From there, each League has two Divisions. For the sample data sets provided, I have one team per Division and a set of 7 players per team. The Teams and Players are identified using an auto incrementing Integer to assure Primary Keys are easily defined and never duplicated.

The Players are assigned a position based on a relation to the Position table. This table includes all major positions in Baseball, as well as an Unassigned field for special cases.

Each team has an associated Stadium. That Stadium has a separate table for its staff members, which I have populated a staff member for each location.

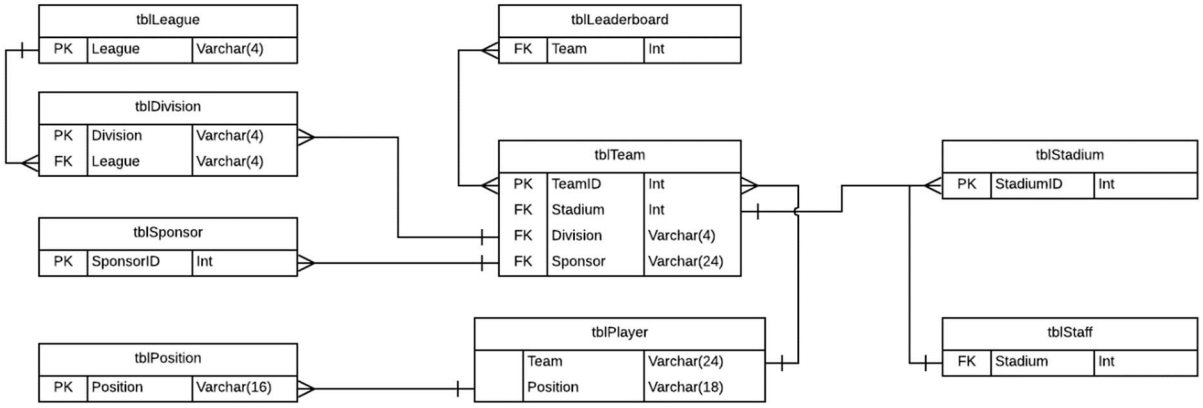
Finally, each team has an associated sponsor. Teams are allowed to have no sponsorship however, but I have created a method to randomly assign a sponsor from the sponsor pool if the team were to request it.

This database follows Boyce-Codd Normal Form has nine entity sets with eight functional dependencies.

Database Systems Project
March 31st, 2019

Alexander Hurley

Part 2: Relational Schema



The Relations: Above I have provided a stripped down version of the ER Diagram for reference. This version only shows the relations and how they all interact with each other.

Let's break the relations down on a per-table basis.

tblLeague: The primary key 'League' in this table is the foreign key in the Division table. This key is very important as much of the database can trace itself back to this key.

tblDivision: There are two relations in this table. One is the foreign key 'League' coming from tblLeague. Each Division has to be part of a larger league in the case of this database. The second relation is the primary key of this table, 'Division'. The Division is the next part of the leagues hierarchy, further breaking down the teams. The Division relates directly to tblTeam.

tblTeam: This table contains the most relations out of the database. The 'TeamID' primary key relates to tblLeaderboard for tracking scores and relating them to their associated team. 'TeamID' also relates to tblPlayer, where it associates players to their respective team. 'Division' as noted before is how each team relates to its respective Division and in turn, how it relates to its overarching League. The 'Stadium' foreign key relates to tblStadium, which associates to the 'StadiumID' field of that table. Finally, the 'Sponsor' field relates to whichever sponsor the given team associates with. This field can be blank if the team does not currently have a sponsor.

tblPlayer: This table has two relations. The first being the 'Team' Field associating each player with their team. The second one is the 'Position' field which relates back to the list of positions each player can have.

tblPosition: This table has one relation connecting the list of positions with the 'Position' field in tblPlayer.

tblLeaderboard: This table has one foreign key relation, and that is to associate the wins and losses back to the corresponding teams.

tblSponsor: This table also has one relation, being the 'SponsorID' field which relates to the Teams directly.

tblStadium: This table has two relations. The first is the relation between 'StadiumID' and the team that is associated with the stadium. The second is 'StadiumID' relating back to tblStaff so that staff members can associate with their given Stadium of employment.

tblStaff: This table has one relation associating it back with tblStadium so staff members can be directly connected with the stadium of their employment.

schema Provided below for reference, as well as in a .sql file

```
CREATE TABLE tblLeague (
League VARCHAR(4) NOT NULL PRIMARY KEY,
DH BOOLEAN NOT NULL
);

CREATE TABLE tblDivision(
Division VARCHAR(4) NOT NULL PRIMARY KEY,
League VARCHAR(4) NOT NULL,
FOREIGN KEY (League) REFERENCES tblLeague(League)
);

CREATE TABLE tblSponsor(
SponsorID INTEGER PRIMARY KEY AUTOINCREMENT,
SponsorName VARCHAR(24) NOT NULL UNIQUE,
Street VARCHAR(36) NOT NULL,
City VARCHAR(24) NOT NULL,
Province VARCHAR(2) NOT NULL,
Postal VARCHAR(6) NOT NULL,
Country VARCHAR(2) NOT NULL
);

CREATE TABLE tblPosition(
Position VARCHAR(16) NOT NULL UNIQUE PRIMARY KEY
);

CREATE TABLE tblStadium(
StadiumID INTEGER PRIMARY KEY AUTOINCREMENT,
StadiumName VARCHAR(36) NOT NULL,
Street VARCHAR(36) NOT NULL,
City VARCHAR(24) NOT NULL,
Province VARCHAR(2) NOT NULL,
Postal VARCHAR(6) NOT NULL,
Country VARCHAR(2) NOT NULL,
Seats INTEGER NOT NULL,
Indoor BOOLEAN NOT NULL
);

CREATE TABLE tblPlayer(
PlayerID INTEGER PRIMARY KEY AUTOINCREMENT,
First VARCHAR(24) NOT NULL,
Last VARCHAR(24) NOT NULL,
DOB DATE NOT NULL,
Team INT NOT NULL,
Position VARCHAR(18),
Injury BOOLEAN,
HandDominance VARCHAR(1) NOT NULL,
Nationality VARCHAR(3) NOT NULL,
Hometown VARCHAR(24) NOT NULL,
BattingAVG INTEGER(1000) NOT NULL,
FOREIGN KEY (Team) REFERENCES tblTeam(TeamID),
FOREIGN KEY (Position) REFERENCES tblPosition(Position)
);

CREATE TABLE tblStaff(
StaffID INTEGER PRIMARY KEY AUTOINCREMENT,
First VARCHAR(24) NOT NULL,
Last VARCHAR(24) NOT NULL,
DOB DATE NOT NULL,
Role VARCHAR(24) NOT NULL,
Stadium INTEGER NOT NULL,
```

```
FOREIGN KEY (Stadium) REFERENCES tblStadium(StadiumID)
);

CREATE TABLE tblTeam(
TeamID INTEGER PRIMARY KEY AUTOINCREMENT,
Team VARCHAR(24) NOT NULL,
Stadium INTEGER NOT NULL,
Division VARCHAR NOT NULL,
Sponsor INTEGER,
FOREIGN KEY (Stadium) REFERENCES tblStadium(StadiumID),
FOREIGN KEY (Division) REFERENCES tblDivision(Division),
FOREIGN KEY (Sponsor) REFERENCES tblSponsor(SponsorID)
);

CREATE TABLE tblLeaderboard(
Team INTEGER NOT NULL UNIQUE,
Playoff BOOLEAN NOT NULL,
Wins INTEGER,
Losses INTEGER,
FOREIGN KEY (Team) REFERENCES tblTeam(TeamID)
);
```

Database Systems Project Alexander Hurley
March 31st, 2019

Part 3: Loading data into SQLite

I have provided 4 .sql files along with this document.

They provide the following:

create_tables.sql – Creates the tables used for this project

drop_tables.sql – Drops tables and clears the auto increment data

populate_tables.sql – Populates all sample data into the database

reload_tables.sql – Drops all tables, recreates them, and populates them for your convenience

Part 4: SQL Interaction – Data Selection

queries Provided below for reference, as well as in a .sql file

The first query here simply pulls the names and birthdate of all injured players and returns the values.

```
/*List Injured Players*/
SELECT tblPlayer.First, tblPlayer.Last, tblPlayer.DOB
        FROM tblPlayer
        WHERE Injury=1
        ORDER BY Last ASC;
```

The second and third queries here return the teams and wins/losses for teams in the playoffs and all teams respectively.

```
/*List Teams in Playoffs*/
SELECT tblTeam.Team, tblLeaderboard.Wins, tblLeaderboard.Losses
        FROM tblLeaderboard
                INNER JOIN tblTeam
                ON tblTeam.TeamID=tblLeaderboard.Team
        WHERE Playoff=1
        ORDER BY tblLeaderboard.Wins DESC;
```

```
/*List All Teams in Leaderboard*/

SELECT tblTeam.Team, tblLeaderboard.Wins, tblLeaderboard.Losses

        FROM tblLeaderboard

                INNER JOIN tblTeam

                ON tblTeam.TeamID=tblLeaderboard.Team

        ORDER BY tblLeaderboard.Wins DESC;
```

If you look at the League table of the database, you can see that there is a Boolean field title 'DH'. This is because one of the Leagues can have a designated hitter, and the other cannot. This next query will find designated hitters who are in a league that does not allow it. This is so you will know who needs to be traded or reassigned to different positions.

```
/*Selects and displays all players who are Designated Hitters in a league where Designated Hitters are
not allowed.*/

SELECT tblPlayer.First, tblPlayer.Last, tblTeam.Team, tblDivision.Division

        FROM tblLeague

                INNER JOIN tblDivision

                        ON tblLeague.League = tblDivision.League

                INNER JOIN tblTeam

                        ON tblDivision.Division = tblTeam.Division

                INNER JOIN tblPlayer

                        ON tblPlayer.Team = tblTeam.TeamID

        WHERE tblLeague.DH = '0'

        AND

        tblPlayer.Position = 'Designated Hitter'

        ORDER BY tblPlayer.Last ASC;
```

For the fifth query, we will list the roster for a specific team. In the example below, we are using Team 2.

```
/*Lists the roster for a specified team*/
SELECT tblPlayer.First, tblPlayer.Last, tblPlayer.Position, tblPlayer.HandDominance
        FROM tblPlayer
                INNER JOIN tblTeam
                ON tblTeam.TeamID = tblPlayer.Team
        WHERE tblTeam.TeamID ='2'
        ORDER BY tblPlayer.Position ASC;
```

For the final query, we will count the number of players from outside of Canada and group by each team.

```
/*Counts the number of players from outside of Canada from each team*/
SELECT tblTeam.Team, COUNT(tblPlayer.PlayerID)
        AS NumberOfPlayers FROM tblPlayer
                INNER JOIN tblTeam
                ON tblTeam.TeamID = tblPlayer.Team
        WHERE tblPlayer.Nationality != 'CAN'
        GROUP BY tblTeam.Team;
```

Database Systems Project Alexander Hurley
March 31st, 2019

Part 4: SQL Interaction – Data Modification

queries Provided below for reference, as well as in a .sql file

The first modification will update the Leaderboard table with the new win and loss for each team from a recent game. In this example we are using Teams 1 and 2.

```
/*Update the leaderboard with new wins or losses*/

 /*Teams 1 and 2 are examples in this statement*/

 UPDATE tblLeaderboard

        SET Wins = Wins + 1

        WHERE Team = '1';

 UPDATE tblLeaderboard

        SET Losses = Losses + 1

        WHERE Team = '2';
```

The second modification will zero out the leaderboard at the end of a season. This will zero out the wins, losses, and remove any teams from the playoffs.

```
/*Zeroes Out the Leaderboard at the end of a season. There is no WHERE clause as I am zeroing
everything. */

UPDATE tblLeaderboard

        SET Wins = 0;

UPDATE tblLeaderboard

        SET Losses = 0;

UPDATE tblLeaderboard

        SET Playoff = 0;
```

The third modification will remove all players older than 50 with injuries from the Player pool.

```
/*Removes Players Older than 50 Years With Injuries*/


DELETE FROM tblPlayer

WHERE DOB <= '1969-01-01'

AND Injury = '1' ;
```

The fourth modification will automatically assign a sponsor to a team if they do not have one. This will randomly pull a sponsor from the sponsor pool.

```
/*Automatically assigns a sponsor randomly if a team in the League does not have one*/

UPDATE tblTeam

        SET Sponsor=

                (SELECT SponsorID

                        FROM tblSponsor

                        ORDER BY RANDOM()

                                LIMIT 1)

        WHERE Sponsor ='';
```

The fifth and final modification will update null positions for players with the ‘Unassigned’ tag.

```
/*Updates Unnasigned as Player Position if null*/

UPDATE tblPlayer

        SET Position = 'Unassigned'

        WHERE Position = '';
```