CS3200 Final Project: MLB Repository
Justin Chen
4/29/2022

README
This database stores MLB players, teams, leagues as well as their stats. The application allows for basic data exploration as well as player management.

Database: MySql and Workbench

Front-end Languages and Libraries: Python 3.10 (Install with instructions on Python website), PyMySql (python3 -m pip install PyMySQL), Pandas (pip install pandas)
- All of these must be installed in order to run the application

To run Baseball Reference web-scraper: requests (pip install requests), jupyter notebook (pip install notebook), and BeautifulSoup4 (pip install beautifulsoup4) are required

Run instructions
1) Mlb_repo_final_dump.sql creates the necessary tables, procedures, and triggers within a database 'mlb'
2) 'Python Application' folder houses two files: main.py, utils.py. main.py houses the main functions and utils stores helpers.
    a) Running main would start the application

Project Presentation and Demonstration: https://www.youtube.com/watch?v=4Gc7ejRBejo

Specifications

Database

- MySql Workbench
    - Install with instructions on MySql website

Application and Libraries

- Python 3.10
    - Install with instructions on Python website
- PyMySql (Connector)
    - Install using pip (python3 -m pip install PyMySQL)
- Pandas (used in the back-end code to wrangle and hold tables)
    - Install using pip (pip install pandas)

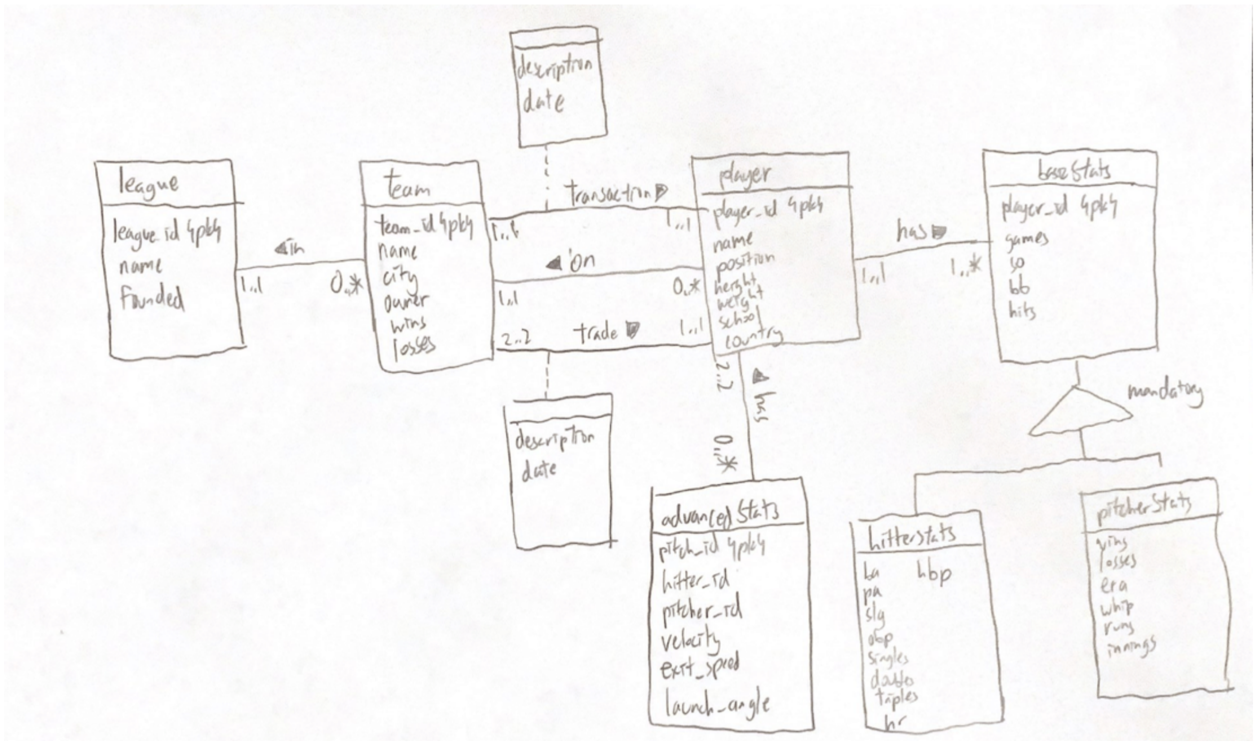
IDE

- Visual Studio Code
    - Download from website

Other

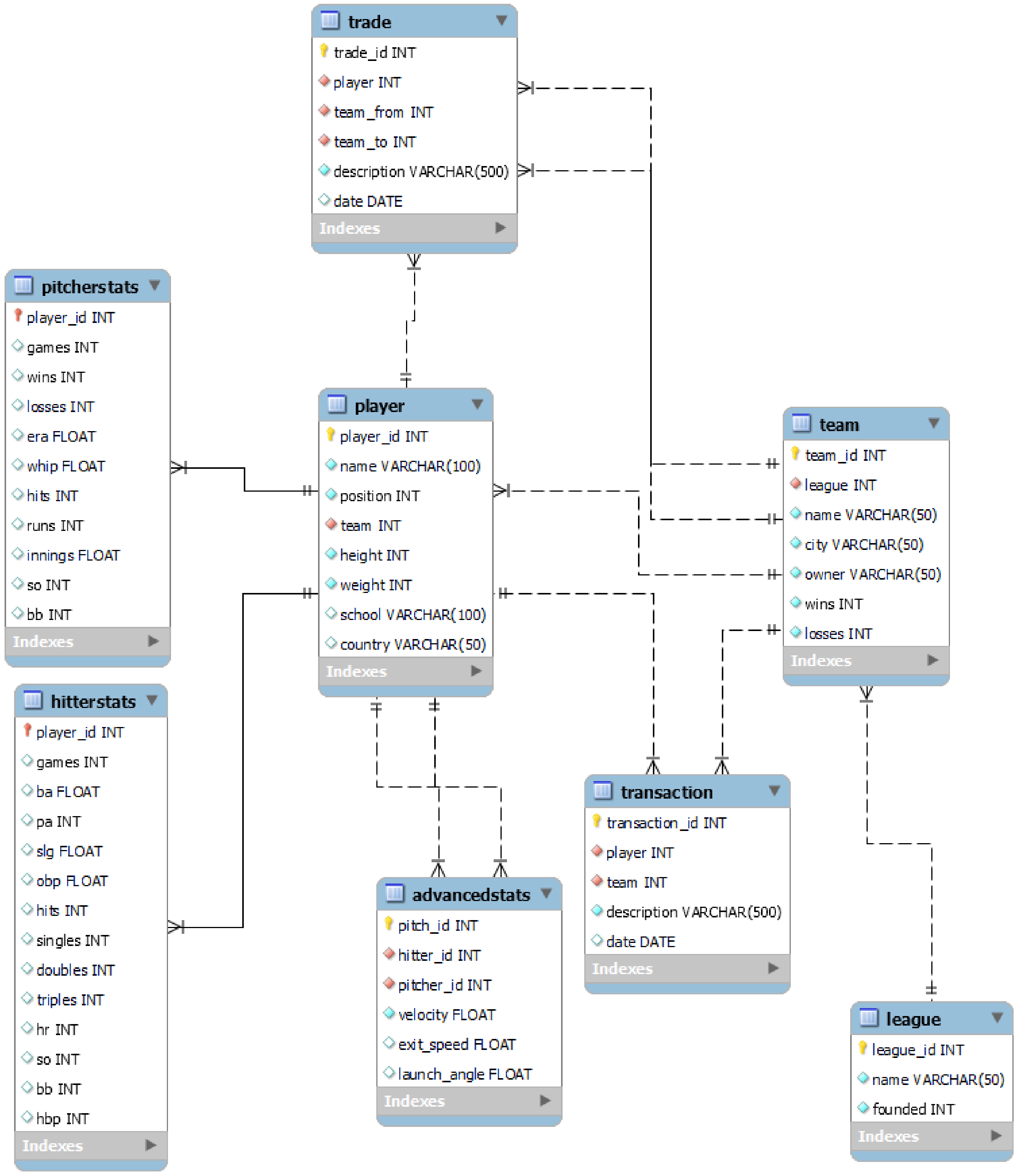
- Sports Reference scraper developed with Python and Jupyter Notebook
    - Install with pip (pip install notebook)
    - Scraper used requests (pip install requests) and BeautifulSoup4 (pip install beautifulsoup4)
- Project was developed on my laptop
- Version control done with Git and Github

UML



description
date
league
league_id 4pk4
name
Founded
◀ in
1..1
0..*
team
team_id 4pk4
name
city
owner
wins
losses
transaction ▶
1..6
1..1
◀ on
1..1
0..*
trade ▶
2..2
1..1
player
player_id 4pk4
name
position
height
weight
school
country
has ▶
1..1
1..*
baseStats
player_id 4pk4
games
so
bb
hits
mandatory
◀ has
2..2
0..*
description
date
advanced stats
pitch_id 4pk4
hitter_id
pitcher_id
velocity
exit_speed
launch_angle
hitterstats
ba
hbp
pa
slg
obp
singles
double
triples
hr
pitcherStats
wins
losses
era
whip
runs
innings

Logical Design



User Flow

On start, user is prompted to choose between the “user” and “admin” views

- I. “user” view has choices to view “player”, “team”, and “league”
  - A. On ”player”, user inputs (and validates) a player name, then chooses between “biography” or “stats”
    1. “biography” gives player information (team, weight, weight, country, etc)
    2. “stats” gives choice between “matchup” or “individual” stats
       - a) “Matchup” asks user to input another player;s name and will return the advanced metrics (pitch velocity, launch angle, exit velocity) of all instances of the players facing off
       - b) “individual” asks user to input “advanced” or “basic”
         - (1) “basic” returns counting stats (wins, losses, batting average, slugging percentage, etc)
         - (2) “advanced” returns aggregated Statcast metrics (pitch velocity, launch angle, exit velocity)
  - B. On “team”, user inputs (and validates) a team name, then chooses between “information” or “stats”
    1. “information” gives team information (name, location, owner, record, team roster)
    2. “stats” asks the user to choose between “basic” or “advanced”
       - a) “basic” returns counting stats (ERA, WHIP, batting average, slugging percentage, etc)
       - b) “advanced” returns aggregated Statcast metrics (pitch velocity, launch angle, exit velocity)
  - C. On “league”, user inputs (and validates) and league name, then chooses between “information” or “standings
    1. “information” gives league biography, including year founded, teams
    2. “standings” gives the league standings of all the teams in it
- II. “admin” view has choices to “add”, “edit”, or “delete” a player
  - A. “add” prompts user to input the name, position, team, height, weight, school, and country of a new player
  - B. “edit” prompts to user to input a player_id, a column/value to change and it’s new value
  - C. “delete” prompts user to input a player_id and deletes that player and relevant tuples from the database
- III. At the final view, the user can choose to restart the program (0) or to end it (1)
  - A. If restart is chosen, go back to step I.

Lessons Learned

<u>Technical Expertise</u>

The biggest technical expertise gained was figuring out how to query data from the MySql database to my front-end code. I had to decide whether I wanted to make a procedure or function inside SQL or have a query to get the tuples that I needed. I took into consideration the data structures and types that I was interacting with for each operation as well as how reusable I wanted my code to be. For example, I was fine using procedures and triggers to add and delete players but for my update function I chose to have a f-string query and data-validation to allow me to use the same code to update rows with different data types (since procedures/functions don't allow an argument take in multiple data-types).

Other things I learned included deciding whether or I wanted to wrangle or aggregate my data within a SQL procedure or in the back-end, which depended on how much I needed to reuse the data I queried and the transformations I did. I also learned how to make an application that was simple but also had enough to help the user understand how to do each step.

<u>Other Insights</u>

Deciding what data-types and restrictions I wanted to use to represent attributes was important, since I needed to make sure what I chose gave me enough flexibility to allow all desired possible values but not overcomplicate an attribute's representation. For example, I chose to represent player positions as either 0 (hitter) or 1 (pitcher). I could've used varChar or text but realized that using 0's and 1's (or a boolean) would suffice and make user interactions easier.

Another challenge I experienced was deciding how to break up all my rows of data into tables and figuring out the best way to implement foreign keys. This was essential to make sure no redundant columns were being added and that it is in 3rd normal form – having the ID system I implemented for my players helped ensure this was possible.

<u>Design</u>

As for the front-end code, I decided to use a command line and simplify the user choices. By presenting less choices per step, it reduces the potential for human-input error. Using the flowchart and user flow notes, the application is easily navigable. I came to this conclusion after using my first version of the application and realizing that too many options per step was overwhelming so I broke down a lot of the steps.

A database design issue I ran into was deciding how I wanted to represent or separate hitter and pitcher stats. Originally, I thought to make them the same table and have certain columns be empty if they were not applicable to the player (or just include stats in the player table itself), but I decided to use inheritance on my UML and separate pitcher and hitter stats. The reason behind this was that it made data import easier if a player ended up accumulating (for any reason) stats of the other player type. It also made querying pitcher and hitter stats for a subgroup, like a team, easier.

Bugs and Broken Code

I have tested a lot of edge cases as well as exceptions and also implemented a lot of data-validation checks to make sure user inputs are consistent with data types and restrictions in the database and application. There are no bugs that I am aware of.

Future Work

This database will be used to store more data scraped from Baseball Reference (basic stats) and downloaded from Baseball Savant for future research projects. The structure and UML design allows easy addition of players and statistics, and the ID mapping system allows for tables and rows to be easily joined.

If I were to expand functionality, it would be in the advanced stats table. There are more columns to be potentially added that would be insightful when analyzing pitches and matchups. Furthermore, more teams and player data will be collected to make the database comprehensive. This expansion would allow for team-matchups to be implemented and allow for more data exploration.