CS 3200 Final Project: NBA Draft Database

Justin Lau and David Malone

README

Necessary files in repository:

- dbdfinalprojdump.sql (SQL database file dump)
- finalprojapp.py (python application)

Software/Libraries:

- MySQL server
- Python version 3.9
- pymysql python library (installed using pip, link: https://pypi.org/project/PyMySQL/)

Installation directory: Just download the zip file and extract it in the home directory. Alternatively, one could also extract the application in the directory they always use for projects

Running the application:

I. First ensure that the MySQL server is running.
II. Run the SQL dump in MySQL to initialize the database.
III. Terminal/Command Prompt command:
    A. UNIX/macOS: python3 finalprojapp.py
    B. Windows: py finalprojapp.py
IV. Follow the prompts to interact with the database.

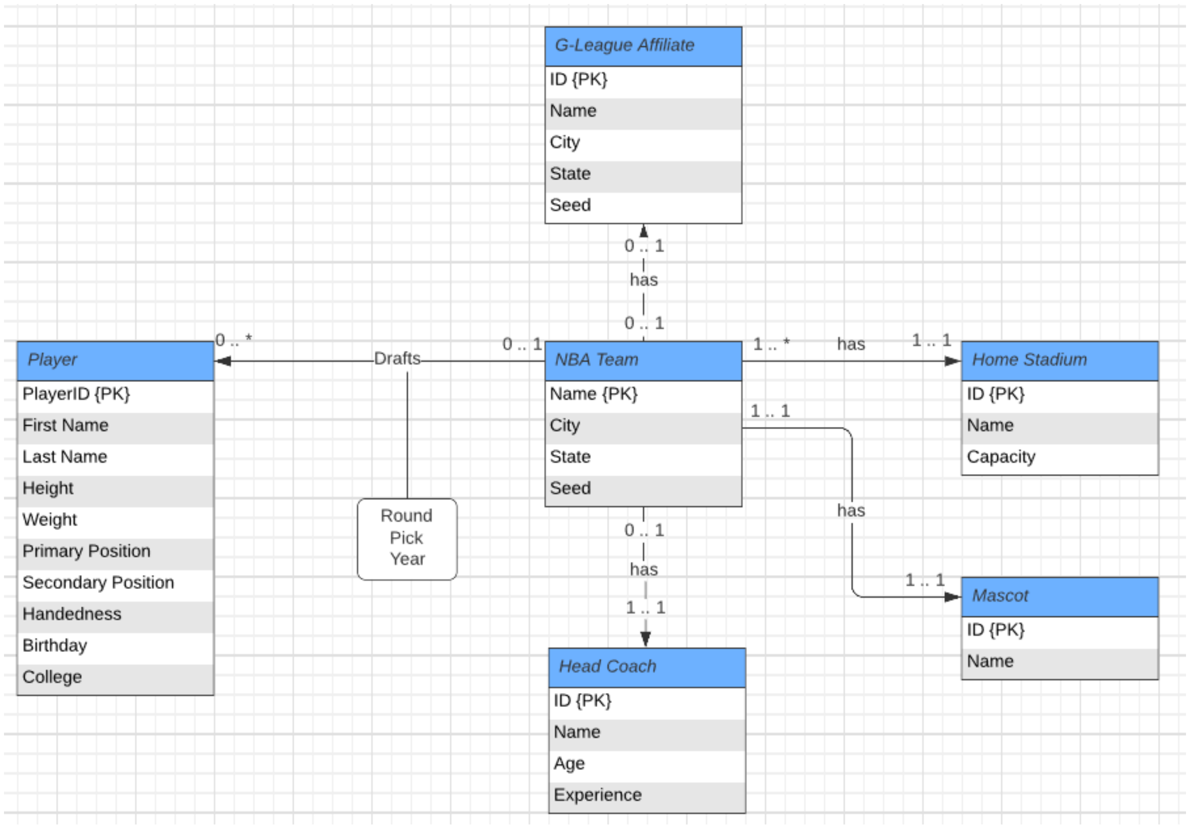
Technical Specifications

Host language: Python3

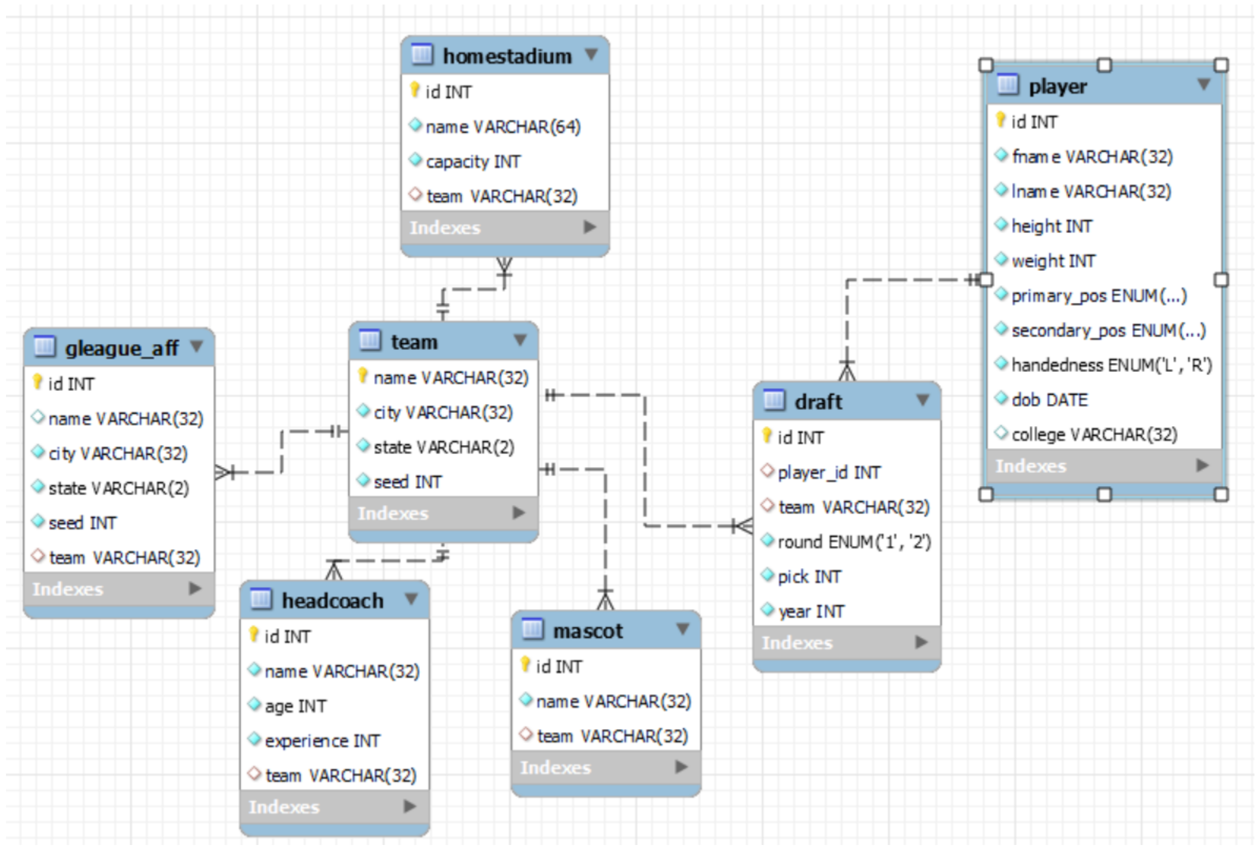
MySQL DBAPI: pymysql

Framework: n/a

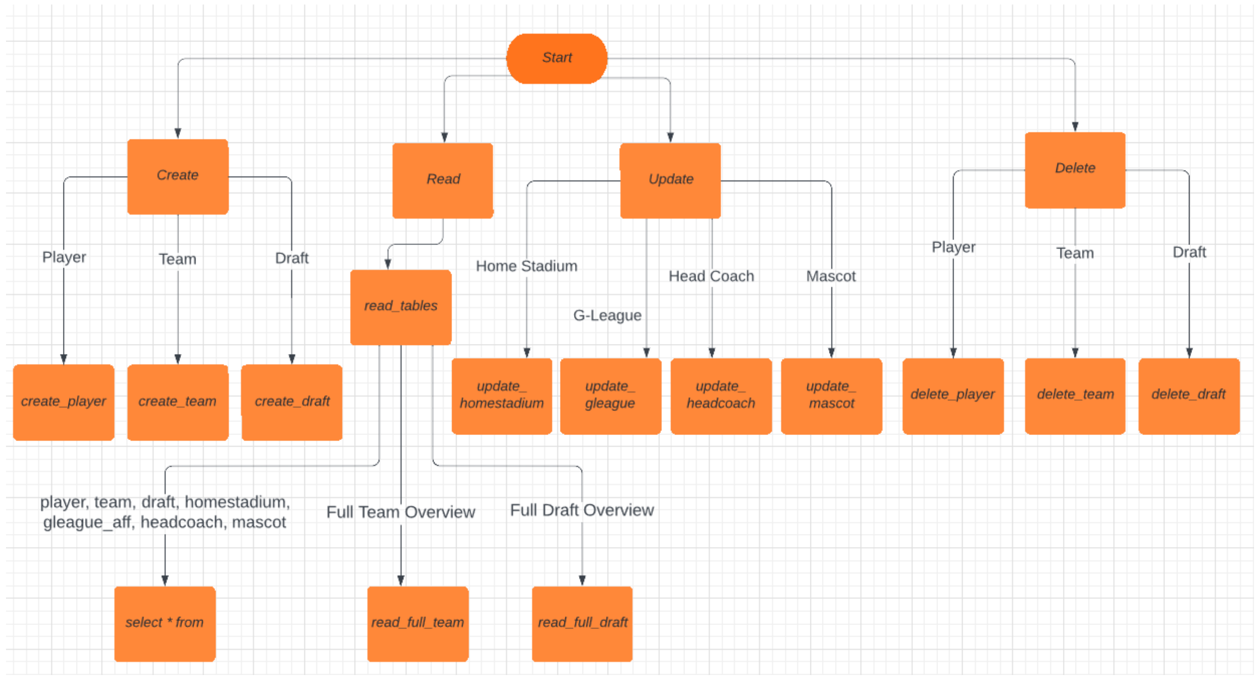
Conceptual Design UML



Reverse-engineered MySQL Schema



User Flow



Lessons Learned

Technical expertise gained:

- Developed practice in programming database applications
- Built upon Python programming skills by parsing and validating user input
- Practiced writing efficient and clean code through helper functions
- Reinforced MySQL skills such as creating tables and procedures, as well as select and insert statements

Insights, time management insights, data domain insights:

- We learned that the more CRUD operations you want to implement, there will be a lot more code to write in the front end and back end. For instance, for one CRUD operation, you will need to write a procedure in MySQL workbench, write a function calling the procedure in Python, and also alter the user input flow in order to call this function. Based on this, we chose the CRUD operations that we felt were most integral to the context of our project (i.e. the most tables affected with the least amount of work).
- Time management was a slight issue since we underestimated the complexity of our project, and as a result spent more time on the back end than expected. As a result, we could not develop a fancier front-end such as a website or GUI.

- We also learned that the NBA has a lot more moving parts than we expected; there is a lot that goes on under the hood in terms of drafting players and building teams!

Realized or contemplated alternative design / approaches to the project:

- Discussed incorporating a website or GUI for our project, but did not have enough time to implement.
- Contemplated designing more CRUD operations, however these operations would not be significant for a user of our database.
- Thought about adding seasons to our database (matches, results, standings), however this would lead to a complex database and perhaps overgeneralize our scope (in addition, some of the functionality would have been covered by draft year)

Future Work

Planned uses of the database:

1. Used to store historical NBA information, and can be used to reflect future drafts in the league
2. Analysis + visualization of player, team, draft stats

Potential areas for added functionality:

1. Conversion to a formal application (i.e. integrate current app with a web framework such as Django, Flask, etc)
2. Web scraping to import existing data into our created tables
3. A more formal graphical user interface (i.e. using ncurses/PyInquierer to have a cleaner-looking TUI; using tkinter to create a desktop GUI; or by virtue, using a web application would give a more formal graphical representation of the app)