Project Final Report

Group Name: GaoLiu
Name: Zihan Liu UID: 001880907
Name: Yuan Gao UID: 001202419

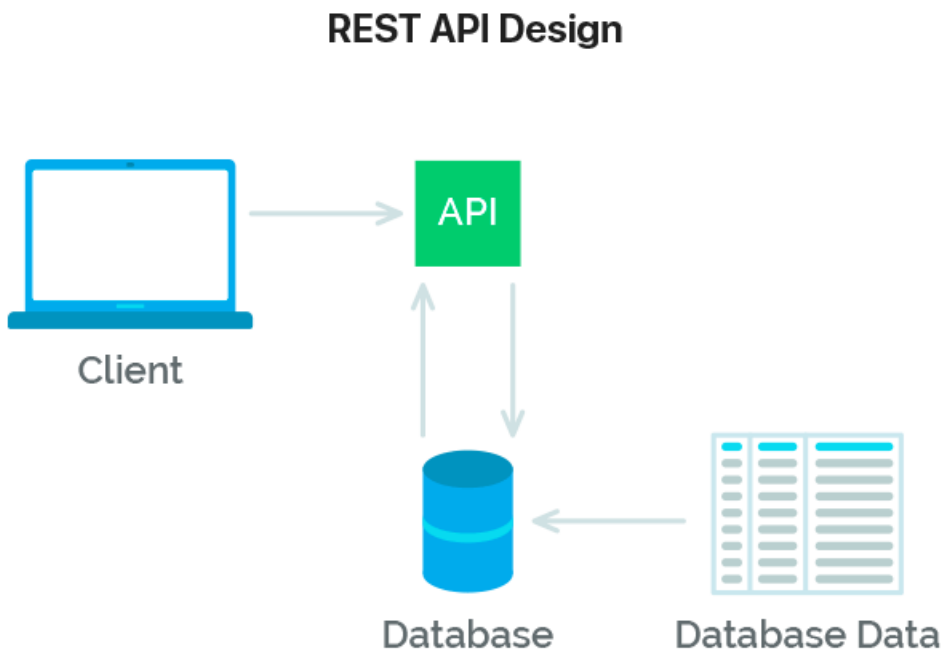
1 Top Level Description

The project is a NBA data statistics website for one specific season. The data domains contain games, teams and players related data. The user can do the following things in our website:

- Display the statistical data of players and teams in one game or average stats in entire season, as well as player and team information.
- Insert new players and new teams.
- Update the information of games, teams and players.
- Delete one specific game, team or player.

2 Architecture

We used the following architecture to implement our project. Basically we use the REST Api architecture.



The advantages of this architecture are:

- Seperate the front end and back end. For us two person group, it is convenient that we can work on one without depending on the other.
- Allows us to run both ends at the same time.
- We can build and deploy both ends together or sepearated.
- Easy to use different tools on each end.

Front-end web framework: AngularJS

AngularJS is a structural framework for dynamic web applications. It has following advantage:

- It provides the capability to create single page application in a very clean and maintainable way.
- It provides data binding capability to HTML. Thus, it gives user a rich and responsive experience.
- Views are pure html pages, and controllers written in JavaScript do the business processing.

Back-end development: Spring Boot and Spring Data JPA

We used spring boot to provide the RESTful api that our frontend needed. Spring Data JPA is responsible for the data access layer, it provides an easy way to access the data in our database. With it we can mapping the data in our database to object that we can manipulate in Spring boot.

It has following advantage:

- It avoids writing lots of boilerplate Code, Annotations and XML Configuration.
- It reduces lots of development time and increases productivity.
- It provides Embedded HTTP servers like Tomcat, Jetty etc. to develop and test our web applications very easily.
- It provides lots of plugins to develop and test Spring Boot Applications very easily using Build Tools like Maven and Gradle

DBMS: MySQL

We choose relational database(SQL) as our main storage. The reason is that our data is structural and well-formed, and many RDBMS have a good balance between durability and performance when doing ACID. For RDBMS, we choose MySQL which is the main database we learned so far.

The schema will be introduced in the following section. We have 8 tables, 8 stored procedures, and 4 triggers.

Procedures:

- get_games_on_date: get all games played on the given date, join 5 tables.
- get_games_given_id: get a game's all information through, join 5 tables.

- get_playes_of_team: get all playes in a team, join 2 tables.
- get_team_of_player: get a player's team info, join 2 tables.
- get_players_given_name: get all players whose name is like given name, used to search player
- get_team_game_desc: get all team games in date descending way, join 2 tables.
- get_player_games_given_team_and_game: get player's stat in a game.
- get_player_game_desc: get all player games in date descending way, join 2 tables.

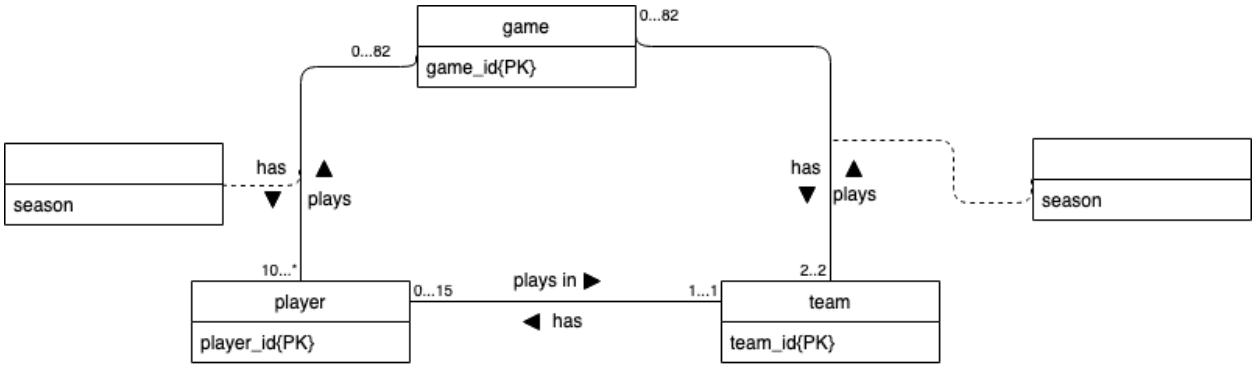
Triggers:

- update_team_season_update: after update on team_game, we need to update the average stats of a given team season.
- update_team_season_before_delete: before delete a team game, we need to update the win-loss info for a given team season.
- update_team_season_delete: after delete a team game, we need to update the average stats of a given team season.
- update_team_game_insert: after insert a game_info, automatically insert two team game for awayTeam and homeTeam.

3 Data Source

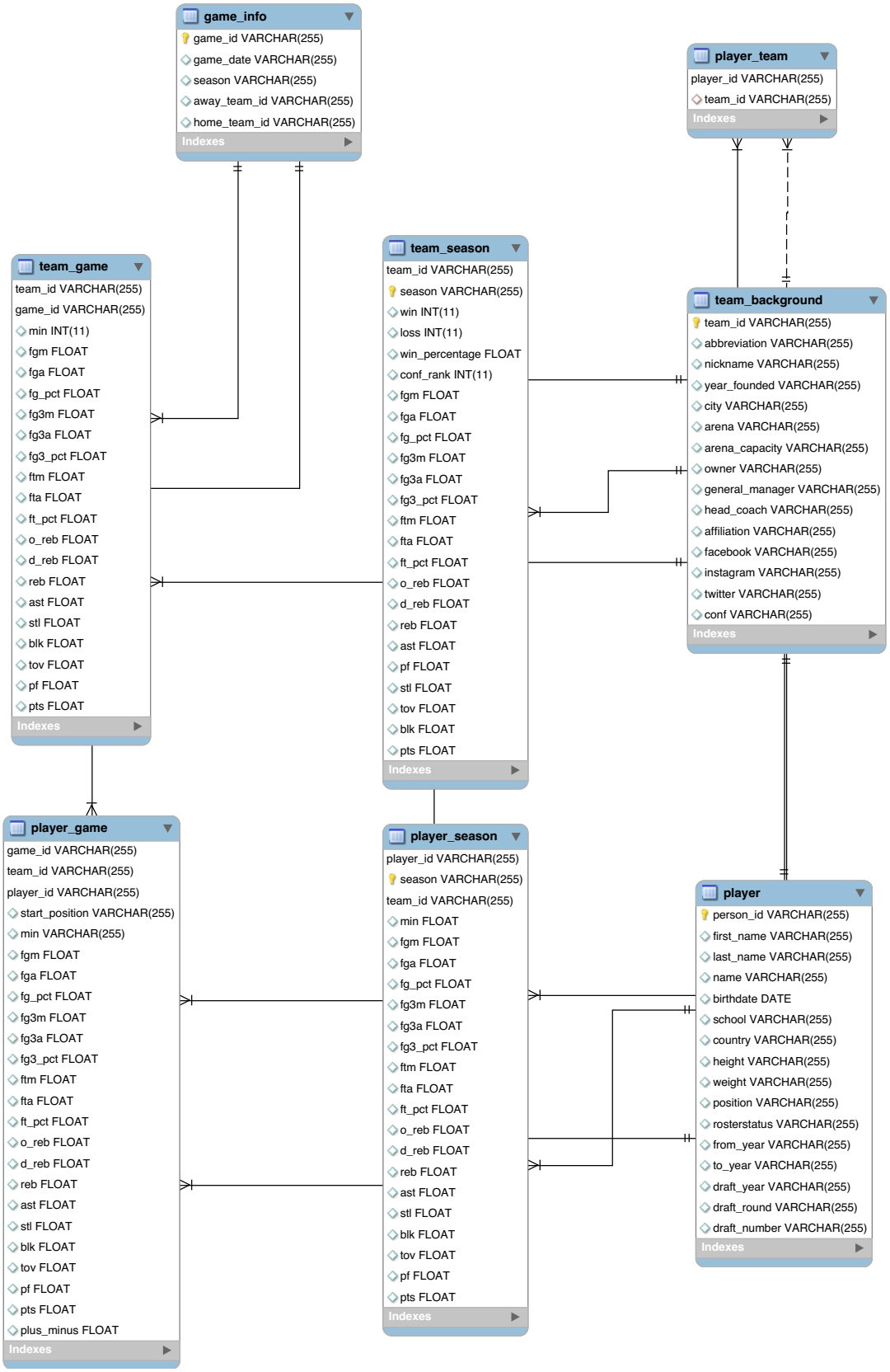
We use NBA_py as our data source. It is a python client for NBA statistics located at stats.nba.com.

4 Conceptual Design



For conceptual design, we can see that we have three main entities, game, player and team. Each team and player will play 0 to 82 games during a season, and one team will have 0 to 15 players, a player will belong to at most one team. During the game, there will be two teams, and no less than 10 players. Based on season and game, we can also derived the average stats of each team and player.

5 Logical Design



6 User flow

7 Lessons Learned

Technical expertise gained

- Use python to fetch and preprocess data.
- Use springboot and spring data JPA to build a restful api.
- Use AngularJS to fetch data from http service and feed to HTML.
- Practice procedure, trigger we learned in course.
- Write complicated subquery.

Group work insights, time management insights, data domain insights

- Good individual and group work. Seperating front end and back end makes the coorporation easier.
- Time management is flexible and not stressful as we start early.
- Data domain is not too complicated but the relationships between game, team and person need carefully think, as our table must follow 3rd and 4th norm.

Alternative design/approaches to the project

- If we can use NoSql, we can store many other information type, including player picture, team picture.
- Backend can use nodeJS and MongoDB, frontend can use Angular. Which is typically a MEAN stack. But this require more time for us to communicate as the application works as a whole.

Function not working

- No function is not working. The data may lack because the player game data is too large and we don't have too much time to fetch and feed them into database. So we just download player games for two team in one season.
- We didn't implement insert player game, as one game will have more than 15 records, which will be a disaster for user to type. So we only implement insert team game stats, our trigger will automatically update the team season after all.

8 Future Work