Link to project:

Description

The proposed application for this project is a comprehensive National Hockey League (NHL) database for game and player statistics from the 2008-2009 season until the current season. The application itself will be a web app focused on getting simple to advanced statistics on players and teams, allowing for an easy user experience when retrieving the data. The aim is to provide a simple way for hockey enthusiasts to look up whatever data they need. The database management system will be designed such that there are administrators who can modify any type of data, team accounts that can modify the data of their team, and users that can simply query the database.

The functionalities that will be supported include searching for individual players and their statistics. This includes stats accumulated in an arbitrary time period, specific games where certain stats were met, and so on. Similar queries will be supported for team searches just with team-specific stats, such as wins between time periods, wins against certain opponents, and so on. An account system with certain saved queries and personalized information will also be supported for the users of the application. This allows certain important and potentially expensive queries to be cached to improve the usability of the app.

System Support

Currently, the application is a web-based app that runs on Windows, MacOS, and Linux currently using local hosts for the frontend and backend sides. This allows for simple launching, testing, and demonstrating of the web app. It would be possible to use a remote server and have the app operate as a more typical website, however, this is an additional step and could realistically only be implemented in later steps, if at all. The programming frameworks chosen are Flask (Python-based web framework) for the backend and Node.js (Javascript-based runtime environment) for the frontend.

The database used in this project will be locally hosted. Again this is for ease of use, testing, and demonstration. It would be possible to host the database on a remote server, and this could be implemented in later milestones. The database will use MySQL as the DBMS to query the database, allow different users to log in, and other database features. Python has MySQL packages which makes the implementation of queries simple on the backend.

Data Gathering

The current plan for getting both toy and sample data is to use and reformat the data held in this NHL, database https://moneypuck.com/data.htm. This database holds detailed statistics from games played from 2009 to the present day. The database also has detailed player information. This database has some redundant information, and its current schema must be modified to match our current schema. The database also uses .csv files which will need to be converted to an appropriate SQL file. However these are minor changes, and the sheer volume of data is exactly what is needed for this project, and this will be the primary source of data.

We will use 3 or 4 players from one season and 3-4 sample teams from the same season for the sample data. This will provide enough information to run the sample queries.

The source used also updates its data regularly, so there must be methods in place to update the database to obtain new data. A solution to this problem is to have the DBMS fetch and update the tables from the source. This could be done manually or scripted to automatically update if the data is too old. This approach could help with the final database, however, it will not be implemented for the sample data as there is plenty of data to use already.

Database Design

Assumptions made about the data being modeled:

1.

Query examples

Searching for players is one feature of the application. In the search bar you can type in a string and any players whose names have a matching substring will appear in the results. The results will be links to the player's page which will then have detailed stats about them. This allows the application to work almost as a search engine. In the future, an algorithm on the backend can detect typos and attempt to return the closest matches. A simple example is case-insensitive searching. Extra clauses can be added to the where section of the query for filtering.

The template query goes as follows. Note that currently, the sample query returns all the attributes associated with the player. This can be modified to only return the necessary attributes, or fancy features can make use of these attributes. Also, note that seasons will be an optional join that will only join on the specified seasons.

```
select *
from Player <optional join with seasons>
where name like '%<searched string>%' <optional filters here>
```

As a sample query:

```
select *
from Player
where name like '%Alex%' and age >= 30
```

A useful feature of this is that with an empty substring, all players will be returned. This allows for finding players based on these optional filters.

Searching for teams is another feature of the application similar to searching for players. The search result will return teams instead. Filters can be used to narrow down teams, however, the filters will now be different. Teams can be viewed as one team, or if a year filter is selected the teams from just those seasons will appear.

The template query goes as follows, again the same nuances apply to the returned attributes.

```
select *
from Team <optional join with seasons>
```

```
    where name like '%<searched string>%' <optional filters here>
```

As a sample query:

```
select *
from Team
where name like '%New%'
```

A player's team will be retrieved and shown on the player's personal page.

A template query goes as follows:

```
select team_name
from
```

On a player page, there will be a section for recent games played. These games will be retrieved from the database. Initially only the 5 most recent games will be shown, subject to the constraints given by the filter from the search. This can be changed later on the add an option to show more games.

The template query goes as follows:

```
select *
from Game
limit 5
```

As a sample query:

Members

- Andrew Wang (20882272). Responsible for the front-end and back-end implementations and implementing queries.

- Shayan Mohammadi Kubijari (20967990). Responsible for schema design, query implementations, and converting the database to SQL, along with modifying the database to fit the schema.

- Arvin Asgharian Rezaee (20972431). Responsible for the front-end and back-end implementations, setting up the scripts to launch the project and query implementation.

- Victor Astinov (20851407). Responsible for the template queries, documentation, and sourcing of the database.

- Francis JIN (21051963). Responsible for the schema design, ER diagrams

All team members did their part as expected. The progress made for all team members is the expected progress for milestone 1.