Project 1 : The NBA Stats Database

COP 4710 Database Design

Due: 11:30pm, Jan 27, 2019

1 Description

In this project, you are expected to build a small in-memory database that stores data related to NBA coaches and teams. Users can send simple queries to this database and add new data to the database. The database has two tables (or relations)—one for coaches and one for teams. The schemas for the two tables are as follows:

coaches	Coach_ID :	7 or fewer capital letters followed by 2 digits,
	season :	4 digit year,
	first_name :	any reasonable English name,
	last_name :	any reasonable English name,
	season_win :	non-negative integer,
	season_loss :	non-negative integer,
	playoff_win :	non-negative integer,
	playoff_loss :	non-negative integer,
	team :	capital letters and/or digits

teams	team_ID :	capital letters and/or digits,
	Location :	American city name (one or two words),
	Name :	team name,
	League :	one capital letter

You must follow these schemas exactly. Your database should have a command-line interface to allow users to add data and send in queries. The interface should accept the following commands:

1. `add_coach`: add a new record describing the performance of one coach in one season. It should have the following 9 arguments: ID, season, first name, last name, season wins, season losses, playoff wins, playoff losses, and team, the types of which should match the schema of the "coaches" relation above;

2. `add_team`: add a new record with one team's basic information. It should be followed by the following 4 arguments: ID, location, name, and league, the types of which should match the schema of the "teams" table;

3. `load_coaches`: load multiple coach and season records from a file. The argument should be the name of the file to read. The file stores each record in one line of text, and different fields of the line/record are separated by commas. In our sample files, you may find empty attribute values (nothing in between two commas) and you should still load that line into your database instead of rejecting it.

4. `load_teams`: load multiple teams' records from a file. The argument should be the name of the file to read. Records are organized in the same way as for `load_coaches`.

5. `print_coaches`: print a list of all coaches, with info about one coach's performance in one season in a line;

6. `print_teams`: print a list of all teams, with info about one team per line;

7. `coaches_by_name`: print the information of coach(es) with the specified last name (the only argument);
8. `teams_by_city`: print the information of teams in the specified city (the only argument);
9. `best_coach`: print the name of the coach who has the most net wins in a given season (the only argument). Net wins should be calculated as (season wins − season losses) + (playoff wins − playoff losses);
10. `search_coaches`: print the info of coaches with the specified properties. This command may have multiple arguments (1+). Arguments are given in the format `field=VALUE` where `field` represents the name of a search criterion and `VALUE` is the value of that field you want to return. For example, a command `search_coaches first_name=John season_win=40` should return all performance data for coaches with first name `John` who had 40 seasonal wins. Note that fields should exactly match one of the column names in the coaches table (ignore any that do not match a column in this table).

Note that a coach's last name can be two words with a space in between (e.g., van Gundy). Your program should be able to handle this, and there will be test cases that search by such names. In order to not confuse your program, we separate the two words of the last name using a `+`. Example query: `coaches_by_name van+Gundy`

Obviously, your job here would be to process the argument by replacing the `+` sign with a space before you do the search. You should do the same for city names and other commands such as `add_coach` and `add_team`.

2 Getting Started

To get started, you can download the compressed archive `proj1.tar`. This package contains a Java command parser that will help you process the commands and their arguments. If you choose to program in Java, the whole command line interface is there for you to use—you only need to define data structures and code the individual commands.

To use this java package, unzip it and type

```
javac *.java
```

to compile the java code. If everything compiles, you can type

```
java P1
```

to execute the program. At this moment, you will see a prompt in your console, and you can now test your database commands. If you choose to program in another language, your program should work in the same way (once the program is executed, it always waits for a command input).

The package also contains two sample input files: `teams.txt` and `coaches_season.txt`. You can take a look at both files to see what kind of data will be used to test your program. Note that both files use commas to separate the attributes in each record. Your database should not store these commas (when using `load_coach` and `load_team`). Furthermore, data records in the file `coaches_season.txt` contain one attribute that we do not use in our schema—the third attribute (`yr_order`), which has single-digit values. You should ignore this attribute when loading and/or printing coach lists. Our test cases for `load_coach` and `load_team` will use files with the same format as `teams.txt` and `coaches_season.txt`.

3 Programming Environment

You can choose any language you like to finish this project. However, since all the grading will be done on the machine `element.csee.usf.edu`, you should make sure your code runs smoothly on this machine before submission. Instructions to log into element will be added to on Canvas.

4 Grading

Your grade on this project is determined by the number of test cases you pass. We will use 12–15 test cases in our grading, and your grade will be based on how many test cases your code passes. Early next week, we will post some sample test cases for you to check your code while programming.

5 Submission

Write a README in plain text explaining how to compile your code and anything we should know about your code. Put the README and all of your source files in a directory named "proj1". Compress the whole directory by typing (in UNIX machines)

```
tar cvf proj1-xxx.tar proj1
```

where `xxx` is your NetID. This will generate a file named `proj1-xxx.tar`. You will lose 10 points if you do not follow the above convention in compressing and naming your file.

You should submit this file by 11:30pm on Monday, January 27, 2019. Please submit as early as possible, as you may have to spend some time figuring out how the submission link works.

6 Miscellaneous

This project description is subject to change, but only in ways that will make your job easier.

You can assume the database system you are building works in-memory only. In other words, you do not have to write your data into a file.

7 Acknowledgements

Data used in this project is provided by `basketballconference.com`.