# COP 4710 Databases, Fall 2011

# Project 1 : The NBA Stats Database (Due: 09/09/2011)

## 1. Description

In this project, you are expected to build a small 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 (and you must follow the schemas in this project):

**coaches** (*Coach_ID* : consists of less than 7 capital letters and two 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 English word(s),
*Name* : team name, any reasonable English word,
*League* : one capital letter)

Your database should have a command-line interface to allow users to add data and send in queries. The interface accepts 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_WIN SEASON_LOSS PLAYOFF_WIN PLAYOFF_LOSS TEAM, the types of which should match the schema of relation "coaches" mentioned 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 LEAGUE, the types of which should match the schema of the "teams" table;

**3. load_coaches** : bulk load of multiple coach/season records from a file specified by the only argument of the command. Note that the file stores each record in one line of text, and different fields of the line/record are separated by commas.

**4. load_teams** : bulk load of multiple team records from a file specified by the only argument of the command. Records in such files are organized in the same way as in those 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, which is given by the only argument of this command;

**8. teams_by_city** : print the information of teams in the city specified by the only argument;

**9. best_coach**: print the name of the coach who has the most net wins in a season specified by the only argument. Note that the net wins should be calculated as (season_win - season_loss) + (playoff_win - playoff_loss).

**10. search_coaches**: print the info of coaches with the specified properties, which are given by the arguments in the following format: *field*=VALUE where *field* represents the name of a search criterion and 'VALUE' is the value of that field you want the query results to match. Multiple fields can be used in the same query. For example, a command "search_coaches *first_name*=John *season_win*=40" means "finding all performance data of a coach with first name 'John' who had a seasonal win of 40". Note that a meaningful field should match exactly one of the column names in the **coaches** table (just ignore those that do not match any column names).

A coach's last name can be two words with a space in between (e.g., van Gundy). Your code should be able to handle this. There will be testcases that search by such names. In order to not confuse your program, we will add a "+" sign between the two words of the last name in the testcases. For example,

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. The same will be done for city names.

## 2. Getting Started

You can download a package (a compressed file named proj1.tar) before getting started. This package basically contains a Java command parser that will help you process the commands and their arguments. In other words, the whole command line interface is there for you to use - you only need to define data structures and code the individual commands. We believe this will greatly reduce your workload in this project if you decide to code in Java.

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 "," to separate the attributes in each record. You should get rid of them in loading the files into your database (e.g., 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 4th attribute with single-digit values. You should ignore them in loading and/or printing coach lists. Our testcases for load_coach and load_team will use files with the same formats as those in teams.txt and coaches_season.txt.

## 3. Programming Environment

You can choose a language you like to finish this project. However, since all the grading will be done in a departmental UNIX machine name netcluster.cse.usf.edu, you need to make sure your code can run smoothly in netcluster before submission. Please consult your undergraduate advisor if you do not have an account in netcluster.

## 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. Passing all test cases means you will get 100 points. Early next week, we will post some sample test cases for you to check your code while programming.

We understand that some minor errors can make your code fail in all (or most) test cases. As our courtesy, one can ask for regrading after modifying no more than 3 lines of code and taking a 15% penalty. There is only ONE chance for regrading. Please contact the TA for regrading requests and all other questions related to this project.

## 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 file and all other relevant files under a directory named "proj1" (remove all data files and binary files, though). 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. If you do not follow the above convention in compressing and naming your file, 10 points will be deducted. You are required to submit this file via the **assignment**

**link** in BlackBoard by 11:30pm, September 9, 2011. Late submissions will cost you free token(s). And remember, you only have THREE tokens for the whole semester! 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 towards a direction 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.

## Acknowledgements

Data used in this project is provided by basketballconference.com