# **CSCI X370: Database Management**

## **Relational Algebra**

Implement the following relational algebra operator: Join (only cases of natural and equi-join, not theta or outer join), Select and Project.

Some of the operators are partially implemented in Table.java (at ELC) and what you need to do is to fully implement those operators. Tuples are stored in comparable lists (ArrayList). You can see some test cases in the main function of Table.java; however, since functions are partially implemented, it will not give the proper output. So, you should be able to see the desired output after the completion of implementation. Download the starter source code from the ELC and compile it. Upon making sure that it is working properly, you can start implementing. If you compile and run MovieDB.java, you will see the tables retrieved based on the queries, which uses partially implemented operators.

Every operator has a method to be implemented and those methods have other methods to be called and to be implemented. So, it may require implementing more than one method to complete an operator. While implementing, also pay attention to the domain (data types) since this could be important to be able to implement some of the operators. The methods may have some parts to be deleted or uncommented right after the implementation; so, please read comments within the code.

**Join** combines tuples from two tables. Furthermore, you need to make equality comparisons per the conditions given in the query (postfix).

Use a (Java) TreeMap for an index.

Your program must be thoroughly documented (generate javadoc). Use the @author tag for each class and method. Each method should have a single author. The coding workload should be split roughly among team-members. We will check this by examining the @author tags and peer evaluations. Please make sure that the output of your program is easy to understand. Provide a flag for turning on/off your tracing/debugging messages in your program's output – if necessary.

**Files Description:**

1. **Table.java:** This file contains Table class which implements relational database tables. Five basic relational algebra operators are provided: project, select, union, minus and join as functions (partially implemented). The insert data manipulation operator is also provided. Missing are update and delete manipulation operators (you don’t need to implement these at this time). **Implement the code in this file related to join only and where you find “// TO BE IMPLEMENTED”.**
2. **Keytype.java:** This file contains KeyType class which provides a key type for handling both non-composite and composite keys. A key is a minimal set of attributes that can be used to uniquely identify a tuple. Do not make any changes to file.
3. **ArrayUtil.java:** Concatenate two arrays to form a new wider array. Do not make any changes to this file.
4. **MovieDB.java:** This file contains MovieDB class which generates a Movies database, which is used to evaluate the relational algebra operator join implemented by you in Table.java class. **You can comment out test cases which are not related to join in this file.**

**Rubric:** Please find the rubric for point distribution and project expectations.

**Programming language:** Java is required for the project.

**What to submit:** Please submit

* all source code
* all the javadoc files
* a readme file

The readme file should contain: your names, how to compile and run your code and other specifications you want to make. Please pack all your files in a zip package with the file name: "project1\_groupName”. For example: project1\_dbDesigners.zip

**How to Submit:**

Submit your ".zip" file using ELC. Only team leaders need to make a submission. **Every student needs to submit a peer-evaluation within 24 hours of the project submission deadline** at ELC. 10 points penalty will be applied to a score in case of a missing peer evaluation.

Do not place your solution on a public web site. Submit your own work and follow the course misconduct policy.