CSC 173 – PROJECT 4 WRITE UP

Project members: Thu Hoang Phan Ha (thoangph), Jiahao Lu (jlu39), Ye Na Kang (ykang19)

Part 1:

1. For this project, we decided to go for the generic implementation from the beginning. We decided to have a struct called Database containing all relations, a Relation struct, and a Tuple struct to contains all the information. Each relation is an array of LinkedList of Tuples, where the index of the array is the hash key. We arbitrarily take the first column in the relation to be the primary index and put the tuples in the Relation table based on that. The relation also holds information about its schema and the size of the schema. Each tuple holds an array of strings that contains the information of one row in the table. After that, we created the constructors for relation, tuple and created 5 tables CSG, SNAP, CDH, CR, CP as described using those generic methods.

2. For the insert, lookup and delete implementations, we also tried to create 3 generic methods:

- Insert takes in a Tuple and a Relation. The method tries to hash the first element in the tuple and put it in corresponding space in the relation
- Lookup takes in a query string and a Relation. We can look up any attribute of any table using this function and create a new Relation based on the user's prompt.
- Deletion takes in a query string and a Relation. It works similarly like lookup and would return a Relation with removed tuples.

3. We used insert to populate the Relations with the data given in the main method.

4. For readfile method, it takes in a filename, which should contain all the relation tables in it. By reading the title for each relation, we create a new relation inside the database. Say the pointer sees "CSG", then it will create the relation CSG. For the writefile method, it takes in a database and loops through each relation. Starting from CSG, it prints out every tuple in the relation. Because we print out "CSG" itself, then we decide not to print out the schema. We also decide to leave one line of space between each relation to make it clear and easier to read.

Part 2: We created two methods that answers the query using the needed relations. We implemented the method described in the "Navigating over many relations" in order to do this part.

Part 3:

1. For the selection method, we prompted the user for a query in the form of C=a (i.e Course="CS101"). Then, we split the string, put in into a string array and turn that query array into the form of the lookup function from part 1 for find the rows needed.

2. For the join method, we tried to visit every tuple in each relation, compare the values of the required attributes. If they are the same, insert into the final relation.

3. For the projection method, we tried to extract the element of the attribute, add it into a tuple and create a new Relation with this tuple.

It was pretty tricky for us to choose to implement the Relational Data Model generically instead of hardcoding each table, but we had fun (and pain) doing it.

Thank you for reading ^^