CMPUT 291

Project 2 Design Document

Group ID:

C291g59

Group Members:

Yuezhou Bao

Devin Dai

General overview

This user interface is created with the functionality of performing Boyce-codd normalization given relations in a database, calculating the closure of attributes given user selected attribute set and functional dependencies set as well as checking for equivalence of user selected functional dependency sets. A user may simply follow the instructions given at the various steps to perform the desired tasks.

Component design

First of all, the implementation of this interface consists of three essential sections with each of them corresponding to the three tasks that the interface has to perform with various small helping methods to make the final program function. We believe this is an efficient and secure way of the implementation of these large programs as it allows the programmer to test and check the code easily and also allows the others to be able to understand the flow of data and the program because it is organized in a self-explanatory fashion. For example, in the *functionalDependencySet* class, we have the methods *find_candidate_key*, *find_minimum_key* and *decompose* which perform some of the more crucial parts of BCNF normalization. One part we decided to do was making BCNF normalization a separate python file as it would be too much to include just in one file and we simply imported the contents over to the main file. In terms of the physical side of the interface, it is really straight forward. The user will be greeted with the main page where the desired task can be selected. After that, there are prompts (with instructions) to guide the user through the steps of performing that specific task which will include both selecting given options as well as inputing values. In addition to those classes, we also have some classes for more organized code. For example, class *Schema* creates a schema base class used for inheritance only and class *SchemaBCNF* which creates a class that inherits schema. We have also used python abstract base class for the set up of the interface. Finally, we have our main function which serves to call the other functions (task specific functions) based on the task the user wishes the interface to perform. It is also where we have some of the error and exception handling parts.

Testing strategy

After we have set up the connection between python and SQL, we simply went about testing the various functionalities of the interface task by task. We first tested BCNF normalization by testing it with the data provided in the *InputRelationSchemas* table. For this task, we had to test it parts by parts as the decomposition involves many small procedures which have to work correctly in order to produce the right results. For example, we had to format the input functional dependencies into 2d arrays and we also

have to first get the candidate keys of the relation based on the given functional dependency sets. Then we have to check if the decomposed relations are indeed correct by testing and doing the problems on paper first to see if the computed results match what we have. For testing the attribute closure, we used the data from the *InputRelationSchemas* table by getting the closure by hand and comparing the results. We have also used example from class slides and examples online. For checking the equivalency between two FD sets, we came up with our own test cases aw well as using some example online as we found the given data to be quite limited in terms of giving us an idea on the correctness of our results. In general, we were printing flag values and partial results along the way to find out some of the problems that we encountered.

Group work description

We decided to split the workload in a way such that we are able to complete and test our own parts individually without having to require code from the other person. So Devin Dai was responsible for calculating the attribute closure for the second functionality as well as the implementation of the equivalency between two user selected functional dependency sets. Devin was also responsible for producing the report. While Yuezhou Bao was responsible for performing BCNF normalization for the relations as well as putting together our individual parts and the overall structure of the final program and the quality of the final code. In terms of keeping everything on track, we first decide how the work is splitted and then since the parts are not interconnected like it was in project 1, we settled down on tackling on our parts individually as we would focus better this way. We then update each other on our progress so we would have ideas on the overall progress. As for the hours spent on this project, we would estimate it to be approximately 20+ hours from start to finish.

Other

As for other extra features or functionalities, we had not done anything significantly different such that it is a departure from the scopes and requirements of this project stated on the assignment page.

Additional notes

There is the rule to the interface that it is case sensitive which means that a $\neq$ A and that when entering the table names from the database, the interface expects the input to match exactly what the database has.