**Mini Project 2 Design Document**

**CMPUT 291**

**Lec EB1**

**Brady Pomerleau**

**Stefan Vidakovic**

**General Overview**

As with the previous project, a certain level of object oriented programming was undertaken in the design of this second project. As a consequence of this, the program makes use of two main back-end classes, the Database and Normaliser classes. User interaction is handled in a non object oriented manner and is done in the main script. This simplification of the UI made more sense for this project since there weren’t multiple user accounts/types to differentiate from as before. As one might imagine, the Database class handles any input/output from the connection to the database inputted at the beginning of a program session. Naturally as a consequence of being the intermediary for the database, the Database class receives and transmits data to the Normaliser class and the UI. Similarly, the Normaliser class handles any algorithmic calculation with regards to the database tables, this includes normalization, attribute closure, and functional dependency set equivalence. The Normaliser receivers input from both the UI and the Database and subsequently submits the results of calculation back to the respective other parts.

**Detailed Component Design**

As previously mentioned the calculation of normalization and related calculations are handled by the Normaliser class. In order to perform these calculations, input from the database was first parsed from a string into various data structures, the smallest and most prevalent being the conversion of the list of attributes and the attributes on either side of a functional dependency into its own set. By storing all this information as attribute sets it became trivial to represent the tables as a list as such: [{attribute set},[[{left set},{right set}]...] where the first element of the list is the table’s attribute set and the second element is a list of functional dependencies which are stored as lists with the left hand side of the => operator and right side stored sequentially. Using this data structure we were able to follow the course notes to compute normalization and attribute closure.

The design of the Database class was centred around providing and receiving the necessary information from the Normaliser. In other words, anything that was added to the Database class was done so in correspondence with a developing Normaliser or UI method.

**Testing**

Testing was performed as usual by running the program and checking the state of the table, or the corresponding program output. Naturally this was done with multiple tables from the database.

**Group Work Description**

Brady Pomerleau - The UI, Functional Dependency Set Equivalence, parts of the normalization

Stefan Vidakovic - Attribute closure, parts of the normalization