DATABASE NORMALAIZATION

Group: VARSHINI YENDURI (12615258), MANISHANVITHA POTRU (12608138) , POLA VENKATA SAI VAMSI KRISHNA(12610876) , ARAVIND REDDY MENDU(12613199).

Sample Input:

Given Dataset: The sample dataset contains StudentID, FirstName, LastName, Course, Professor, ProfessorEmail, CourseStart, and CourseEnd. This dataset is composed of five tuples and is stored in a CSV (comma-separated values) file format, making it easy to work with and analyse.

Functional Dependencies: The functional dependencies used for the dataset are:

StudentID->Firstname, LastName.

Course->CourseStart,CourseEnd,Professor.

Professor->ProfessorEmail.

Multi-valued Dependencies: The multi-valued dependencies used for the dataset are:

Course->Coursestart

Course->CourseEnd

The Functional dependencies and multi-valued dependencies are taken as .txt file(input.txt).

User Input: Choice of the highest normal form to read (1NF,2NF,3NF,4NF,BCNF,4NF,5NF).
Primary Key used (StudentID, Course)

Core Components:

The system comprises three core components:

1.Input Parser

2. Normalizer

3.SQL Query Generator.

The Input Parser function reads both CSV and txt files, breaking down text into recognizable character strings for analysis. The Normalizer is responsible for transforming the input dataset into the desired normal form based on provided functional dependencies. Finally, the SQL Query Generator facilitates the generation of SQL queries, streamlining the query creation process.

Results:

Source code: In this context, we employed the Python programming language to perform data normalization on the provided dataset.

Code Description:

Python libraries are imported to facilitate the necessary operations. The input dataset is provided as a CSV file, and the "read" method is employed to extract data from this file. Additionally, the "open" method is utilized to access and read the functional dependencies stored in a text file. User input regarding the desired normal form and primary keys is obtained. The "Parser" method is then applied to convert the input files into strings of characters for further processing.

We conduct various checks to determine the normal form of the given dataset. To start, we verify multi-valued dependencies and establish whether the dataset contains a superkey. The "bcnf_decomposition" function is then employed to decompose the relation into BCNF normal form. We have a series of methods defined for assessing the dataset's normal form status, including "inf," "2nf," "3nf," "bcnf," "4nf," and "5nf" methods. These functions check for INF, 2NF, 3NF, BCNF, 4NF, and 5NF compliance, respectively.

Additionally, we provide functions like "first_normal_form," "second_normal_form," "third_normal_form," "bcnf_normal_form," "fourth_normal_form," "decomposing_to_5nf," and "fifth_normal_form" to decompose the dataset into INF, 2NF, 3NF, BCNF, 4NF, and 5NF, as required.

To generate the desired output results, we define an "output" function. Further, we offer "sql_query_1NF" for INF SQL query output and "sql_query_2_3" for 2NF and 3NF SQL query outputs. Finally, the "sql_query_BCNF_4_5" function is provided to produce BCNF, 4NF, and 5NF SQL query outputs.

All these methods and checks are performed to assess the normal form status of the given dataset, determining whether it adheres to any specific normal form or not.

The highest normal form of the given dataset can be found by using the functions in_1nf, in 2_nf,in_3nf,in_4nf,in_bcnf and in_5nf from the variable high_normalform.

Code Execution:

To normalize the given table, please select the desired normal form by pressing:

1 for INF,

2 for 2NF,

3 for 3NF,

B for BCNF,

4 for 4NF, or

5 for 5NF.

After that, press 1 if you want to obtain the highest normal form of the given relation or press 2 if the highest normal form is not required.

You will then need to enter the primary key values, separated by commas, for this relation. In this case, the primary key should be specified as "StudentID, Course."

If the given table is not already in the specified normal form, the system will decompose the table until it satisfies the chosen normal form, providing the queries for the decomposed tables.

For 5NF, candidate keys are specified for each table. The candidate key for the student table is "StudentID," for the course table it is "Course," and for the professor table, it is "Professor."

Lastly, the highest normal form satisfied by the given table is displayed, which in this case, is only INF.