CS5300 PROGRAMMING PROJECT DATABASE NORMALIZATION

TEAM: Bhavya Shree Kakunuri, Sai Teja Putta

Sample Input:

Dataset: TestingData (1NF-5NF).xlsx, he file "TestingData (1NF-5NF).xlsx" is designed to test the normalization process of a database schema through different normal forms.

Functional Dependencies: The dataset's functional dependencies are as follows:

OrderID --> Date, TotalCost, TotalDrinkCost, TotalFoodCost, CustomerID, CustomerName
OrderID, DrinkID --> DrinkSize, DrinkQuantity, Milk
OrderID, FoodID --> FoodQuantity
CustomerID --> CustomerName
DrinkID --> DrinkName
FoodID --> FoodName

Multi-valued dependencies:

OrderID ->> DrinkID
OrderID ->> FoodID

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

User Input:

1. Select the highest normal form to normalize from the below list 1 for 1NF ,2NF,3NF, BCNF ,4NF ,5NF.
2. Primary key used (OrderID,DrinkID,FoodID)

- Primary keys are taken as user input, allowing flexibility for composite keys. This input is crucial, as primary keys define the relationships and dependencies necessary for each normalization step.

Core Components:

1. Input Parser: The txt file, which is used to contain the functional dependencies, and the Excel file, which is used to contain the table, are read by the parser function. It breaks down the text into recognized strings of characters for further analysis.

2. Normalizer: The input dataset is broken down by the normalizer into the necessary normal form using the specified functional dependencies. The provided dataset is broken down into the user-required normal form using normalizing techniques.

3. SQL Query Generator: It refers to a program or feature that helps in creating SQL queries.

Deliverables:

Source Code: we used Python programming language to normalize the given dataset.

Code Description:

Main Function

The main () function is the driver for the program. It takes as input from the user: the input and output file, the normalization level-a choice between 1NF to 5NF- and the target format, a choice between SQL output and textual schema output.

It then processes the command line arguments to:

1. Check Normal Form: The normal form specified is checked to be within the range of those supported, ranging from 1NF to 5NF.

2. Input Processing: This calls load_data to read in data from the nominated file and then calls load_fds_from_file and load_mvds_from_file, providing parsing of functional and multivalued dependencies, respectively.

3. Primary Key Input: It captures and processes primary key input for later steps in normalization.

4. Normalization Execution: It consecutively calls each normalization function from to_1nf through to_5nf depending on the highest normal form specified. Each successive step in normalization decomposes relations according to its rules and returns a progressively normalized schema.

5. Generation of Output: Produces either SQL statements or a schema representation, based on the target format selected.

Normalization Functions

1. to_1nf: This removes multivalued attributes by splitting them into other relations containing the primary key attributes of the old relation. Then, each tuple from the old relation is expanded for every single multivalued entry, which ensures that there are no more multivalued attributes left.

2. to_2nf: It recognizes partial dependencies and decomposes them, moving dependent attributes into new relations.
This will ensure that dependency will be only on a full primary key, not on subsets of it, which creates 2NF.

3. to_3nf: It recognizes transitive dependencies and then decomposes them. This is performed by the creation of new relations such that in every functional dependency in a relation, there is a superkey or each dependent attribute is prime.

4. to_bcnf: Further refines the schema by eliminating any remaining nontrivial functional dependencies by decomposing relations such that all dependencies have a superkey on the left side as dictated by BCNF.

5. to_4nf: Eliminates multi-valued dependencies by decomposing relations containing multi-valued dependencies into separate relations.

6. to_5nf: Resolves join dependencies by decomposing relations into optimally structured relations that can be joined through natural joins.
This will be achieved through the checking of join dependency and trying the deco mposition until an optimal structure is reached.

Supporting Functions

- load_data, load_fds_from_file, and load_mvds_from_file: Load the main dataset, functional dependencies, and multivalued dependencies from specified files.
- detect_multivalued_columns: Scans for and identifies multivalued attributes within the dataset to prepare them for normalization to 1NF.
- is_superkey (inside to_bcnf): Checks whether a given set of attributes forms a super key in regard to the primary keys.
- check_join_dependency (inside to_5nf): Checks if a relation has join dependency; if so, then decompose into multiple tables for 5NF.

Schema Generation

compile schema builds up a final schema in a user-friendly
format, enumerating each relation and its attributes, along with the primary key. This may support both textual and SQL output, whichever the user may prefer.