Automated Database Normalization Program

Sasidhar Reddy Velkuri -12618248

Venkata Mokshagna Nadella- 12619877

Objective

Develop an automated tool to normalize a relational database schema from 1NF to 5NF. The program will parse user inputs for database schema, functional dependencies (FDs), multi-valued dependencies (MVDs), and data instances where required, providing a final, normalized schema and SQL queries that represent the target normal form.

Requirements and Scope

Input Requirements:

1. **Database Schema (Relations)**
    - **Table Structure**:
        - Table Name and Attributes
        - Primary Keys and Candidate Keys
        - Any Non-Atomic Attributes (multi-valued data)
    - Each table is processed independently, allowing for incremental normalization per table.
2. **Functional Dependencies (FDs)**
    - FDs defined as X -> Y, where X and Y are sets of attributes.
    - The input format allows the program to use FDs to determine normal forms up to BCNF.
3. **Multi-Valued Dependencies (MVDs)**
    - Non-trivial MVDs specified by the user, necessary for 4NF decomposition.
    - Data instances required to verify MVDs, enabling the program to apply MVD-based decomposition only when validation succeeds.
4. **Normalization Target**
    - User-defined target normal form (1NF to 5NF). For example, if the user specifies 4NF, the program will normalize incrementally through 1NF, 2NF, 3NF, and BCNF before performing 4NF decomposition.

Output Requirements:

1. **SQL Queries**
    - SQL CREATE TABLE statements for each normalized table.
    - Include appropriate constraints (e.g., PRIMARY KEY, FOREIGN KEY).
2. **Normalized Schema Representation**
    - Provide a detailed textual or diagrammatic schema showing each normalized table, attributes, and constraints.

Core Components

1. **Input Parser**
    - **Purpose**: Efficiently parse the schema file (tables and attributes), FDs, and MVDs from user-provided files.
    - **Functionality**:
        - Extract table names, attributes, primary keys, and non-atomic data indicators.
        - Parse functional dependencies and multi-valued dependencies from the provided text files.
        - Prepare data for normalization by ensuring relationships and constraints are clearly identified.
2. **Normalization Engine**
    - **Purpose**: Progressively normalize tables from 1NF up to the user-defined target normal form.
    - **Methodology**:
        - **1NF**: Enforce atomic attribute values. Non-atomic attributes trigger new table creation.
        - **2NF**: Remove partial dependencies from tables in 1NF by decomposing based on candidate keys.
        - **3NF**: Eliminate transitive dependencies to ensure that all non-key attributes depend solely on candidate keys.
        - **BCNF**: Decompose any tables with dependencies where the determinant is not a candidate key.
        - **4NF**: Validate MVDs using data instances to confirm applicability, then decompose tables to remove non-trivial MVDs.

        - **5NF**: Identify and remove join dependencies to achieve the highest normalization form if specified.

3. **Validation Engine for MVDs and Join Dependencies**
    - **Purpose**: Ensure that provided MVDs align with data instances, essential for validating 4NF decompositions.
    - **Methodology**:
        - Parse and validate data instances for given MVDs.
        - Verify join dependencies for 5NF (if required) to eliminate redundancy while maintaining data integrity.
4. **Output Generator**
    - **Purpose**: Produce SQL queries and schema representations reflecting the final normalized structure.
    - **Output Options**:
        - **SQL Queries**: Generate CREATE TABLE statements with primary keys, foreign keys, and other constraints.
        - **Schema Representation**: Output a clear, hierarchical schema listing tables, attributes, keys, and dependencies.

Detailed Workflow

1. **Initialization**:
    - Load the database schema, FDs, and MVDs.
    - Parse and organize input data for efficient access.
2. **Normalization Process**:
    - **Step 1: 1NF** - Identify and restructure non-atomic attributes.
    - **Step 2: 2NF** - Decompose relations with partial dependencies.
    - **Step 3: 3NF** - Remove transitive dependencies.
    - **Step 4: BCNF** - Enforce BCNF conditions by ensuring each determinant is a candidate key.
    - **Step 5: 4NF** - Validate MVDs with data, decomposing as needed to satisfy 4NF.
    - **Step 6: 5NF** - If specified, analyze join dependencies and decompose as necessary.

3. **Output Generation**:
    - Based on the target form, generate SQL CREATE TABLE commands with appropriate constraints.
    - Render a detailed schema diagram or table-based textual representation of each table, including key constraints.

Deliverables

1. **Source Code**:
    - Well-documented and modular code for each normalization phase, ensuring ease of maintenance and extensibility.
    - Modules:
        - data_parser.py: Input parsing and data preparation.
        - normalizer.py: Core normalization logic.
        - output_generator.py: SQL generation and schema representation.
2. **Project Documentation**:
    - **Overview**: Describe the overall program structure, purpose, and features.
    - **Methodology**:
        - Describe the approach for each normalization form.
        - Explain handling and validation of FDs and MVDs.
    - **Usage Guide**:
        - Detailed instructions on running the program.
        - Explanation of input formats, customization options, and interpreting the output.
    - **Example Walkthrough**: Step-by-step guide demonstrating a sample database normalization process from input to final output.
3. **SQL and Schema Output Examples**:
    - Examples of normalized SQL queries and schema diagrams, illustrating transformations from raw input to fully normalized output.

Results of Normalization:

Initial Schema:

```
GivenTable
   StudentID FirstName   LastName Section     Contact   Parent_no  \
0          1  Sasidhar    Velkuri       A  5736471122  9035736473
1          2     Pooja  Morampudi       B  5736471122  9035736474
2          3    Moksha    Nadella       C  5736473322  9035736475
3          4      Jhon     Bright       A  5736474422  9035736476
4          5     Lewis   Hamilton       C  5736475522  9035736477

               Streetno     City State  Zipcode Department      Email  \
0            2041 Vichy    Rolla    MO    65401         CS  svdfy@mst
1           2083 Vichy     Rolla    MO    65402        IST  pmkrg@mst
2              13 ozak      Ozak    MP    65403         CS  nvdfy@mst
3  15th apt White Cols    Robert    AZ    65404       MECH  jxbty@mst
4        14th newcastle  Stlouis    MO    65405         CE   l44h@mst

   AdvisorID Advisor name  Advisor Contact
0          1          Ric       5736471462
1          2          Win       5736471462
2          1          Ric       5736473462
3          3         Wang       5736474462
4          4          Das       5736475462
```

Functional Dependencies (FDs):

```
Functionaldependencies
{('StudentID',): ['FirstName', 'LastName', 'Section', 'Contact', 'Paren
t_no', 'Streetno', 'City', 'State', 'Zipcode', 'Department', 'AdvisorI
D'], ('AdvisorID',): ['Advisor name', 'Advisor Contact'], ('StudentID',
'AdvisorID'): ['Section'], ('Zipcode',): ['City', 'State']}
```

Multi-Valued Dependencies (MVDs) & Primary Key's:

```
Choose Higehst Normal form table can reach (1: 1NF, 2: 2NF, 3: 3NF, B:
BCNF, 4: 4NF, 5: 5NF): 5
Find the highest normal form of the input table? (1: Yes, 2: No): 1
Enter Primary keys if it composite enter comma between them: StudentID,
AdvisorID


['StudentID ->> AdvisorID', 'StudentID ->> Section']
MULTI-VALUED Functionaldependencies
{"['StudentID']": ['AdvisorID', 'Section']}
```

Relations After 3NF:

	StudentID	FirstName	LastName	Section	Contact	Parent_no
0	1	Sasidhar	Velkuri	A	5736471122	9035736473
1	2	Pooja	Morampudi	B	5736471122	9035736474
2	3	Moksha	Nadella	C	5736473322	9035736475
3	4	Jhon	Bright	A	5736474422	9035736476
4	5	Lewis	Hamilton	C	5736475522	9035736477

	Streetno	City	State	Zipcode	Department	AdvisorID
0	2041 Vichy	Rolla	MO	65401	CS	1
1	2083 Vichy	Rolla	MO	65402	IST	2
2	13 ozak	Ozak	MP	65403	CS	1
3	15th apt White Cols	Robert	AZ	65404	MECH	3
4	14th newcastle	Stlouis	MO	65405	CE	4

	Advisor name	StudentID	Advisor Contact
0	Ric	1	5736471462
1	Win	2	5736471462
2	Ric	3	5736473462
3	Wang	4	5736474462
4	Das	5	5736475462

Relations After BCNF:

RELATIONS AFTER BCNF

	Zipcode	City	State
0	65401	Rolla	MO
1	65402	Rolla	MO
2	65403	Ozak	MP
3	65404	Robert	AZ
4	65405	Stlouis	MO

	StudentID	FirstName	LastName	Section	Contact	Parent_no
0	1	Sasidhar	Velkuri	A	5736471122	9035736473
1	2	Pooja	Morampudi	B	5736471122	9035736474
2	3	Moksha	Nadella	C	5736473322	9035736475
3	4	Jhon	Bright	A	5736474422	9035736476
4	5	Lewis	Hamilton	C	5736475522	9035736477

	Streetno	Zipcode	Department	AdvisorID
0	2041 Vichy	65401	CS	1
1	2083 Vichy	65402	IST	2
2	13 ozak	65403	CS	1
3	15th apt White Cols	65404	MECH	3
4	14th newcastle	65405	CE	4

```
  Advisor name StudentID Advisor Contact
0          Ric         1      5736471462
1          Win         2      5736471462
2          Ric         3      5736473462
3         Wang         4      5736474462
4          Das         5      5736475462


RELATIONS AFTER BCNF


  Zipcode     City State
0   65401    Rolla    MO
1   65402    Rolla    MO
2   65403     Ozak    MP
3   65404   Robert    AZ
4   65405  Stlouis    MO
```

Final Relation After 5NF:

- **SQL Statements**:

```
OUPUT QUERIES AFTER 5NF:


CREATE TABLE Zipcode_table (
  Zipcode VARCHAR(255) PRIMARY KEY,
  City VARCHAR(255),
  State VARCHAR(255)
);
CREATE TABLE StudentID_table (
  StudentID VARCHAR(255) PRIMARY KEY,
  FirstName VARCHAR(255),
  LastName VARCHAR(255),
  Section VARCHAR(255),
  Contact VARCHAR(255),
  Parent_no VARCHAR(255),
  Streetno VARCHAR(255),
  Zipcode VARCHAR(255),
  Department VARCHAR(255),
  AdvisorID VARCHAR(255)
);
CREATE TABLE StudentID,AdvisorID_table (
  Advisor name VARCHAR(255),
  StudentID VARCHAR(255),
  Advisor Contact VARCHAR(255)
);


Highest Normal Form of input table is: 2NF
```