**Database Documentation**

**Overview**

This tool is designed to analyse and normalize database tables according to various normal forms (1NF through 5NF). It takes a CSV file input containing table data and performs normalization based on user-specified functional dependencies, multi-valued dependencies, and primary keys.

**Features**

- Detects and validates data types

- Analyses table structure up to Fifth Normal Form (5NF)

- Converts tables to user-specified normal forms

- Generates SQL creation statements for normalized tables

- Handles complex dependencies and relationships

- Outputs results to both CSV and text files

**Core Functions**

**Data Type Detection**

check\_datatypes(csv\_filePath)

Analyzes the first row of data to determine appropriate SQL data types for each column.

* Returns: Dictionary mapping column names to SQL data types
* Supported types: INT, VARCHAR (100), DATE, VARCHAR(50)

**Helper functions:**

* is\_integer(attr : Validates integer values
* is\_alphanumeric(attr): Validates string values
* is\_date(attr): Validates date values
* is\_email(attr): Validates email addresses

**Normal Form Analysis**

check\_normal\_form(csv\_filePath, FD, Key, MVD)

Determines the highest normal form of the input table.

Parameters:

* csv\_filePath: Path to input CSV file
* FD: List of functional dependencies
* Key: Primary key attributes
* MVD: List of multi-valued dependencies

Returns: String indicating the highest normal form achieved

**Individual normal form checks:**

* check\_1NF (csv\_filePath): Validates First Normal Form
* check\_2NF (FD, Key): Validates Second Normal Form
* check\_3NF (FD, Key): Validates Third Normal Form
* check\_BCNF (FD, Key): Validates Boyce-Codd Normal Form
* check\_4NF (FD, Key, MVD): Validates Fourth Normal Form
* check\_5NF (FD, Key, MVD): Validates Fifth Normal Form

**Normalization Functions**

convert\_to\_1NF (csv\_filePath)

Converts table to First Normal Form by eliminating non-atomic values.

Returns: Tuple of (non\_atomic\_columns, normalized\_columns)

convert\_to\_2NF (FD, Key)

Eliminates partial dependencies to achieve Second Normal Form.

Returns: Dictionary of decomposed tables

convert\_to\_3NF (FD, Key, tables)

Eliminates transitive dependencies to achieve Third Normal Form.

Returns: Dictionary of decomposed tables

convert\_to\_BCNF(FD, Key, tables)

Decomposes tables to achieve Boyce-Codd Normal Form.

Returns: Dictionary of decomposed tables

convert\_to\_4NF (FD, Key, MVD, tables)

Handles multi-valued dependencies to achieve Fourth Normal Form.

Returns: Dictionary of decomposed tables

convert\_to\_5NF (FD)

Decomposes tables to achieve Fifth Normal Form using join dependencies.

Returns: Dictionary of decomposed tables

**SQL Generation**

generate\_sql\_queries(FD, Key, tables, data\_types)

Generates SQL CREATE TABLE statements for the normalized tables.

**Parameters**:

* + FD: Functional dependencies
  + Key: Primary key attributes
  + tables: Dictionary of normalized tables
  + data\_types: Dictionary of column data types

Returns: List of SQL CREATE TABLE statements

**Usage**

1. Run the script and provide the following inputs:

Enter input file path: [path to CSV file]

Enter Functional Dependencies in the format: A->B; A, B->C; A->B, C

[Enter dependencies separated by semicolons]

Enter Multi Valued Dependencies in the format: A->B; A->C

[Enter MVDs separated by semicolons]

Enter Key:

[Enter primary key attributes separated by commas]

2. Choose whether to analyse the current normal form:

Find the highest normal form of the input table? (1: Yes, 2: No)

3. Select target normal form:

Choice of the highest normal form to reach (1: 1NF, 2: 2NF, 3: 3NF, B: BCNF, 4: 4NF, 5: 5NF)

**Output**

- Normalized tables are saved to CSV files

- SQL creation statements are generated

- Results are written to 'Output.txt'

**Example** **Input**

csv\_filePath = "input.csv"

FD = ["A->B", "B,C->D"]

MVD = ["A->B", "A->C"]

Key = ["A", "B"]

**Check normal form**

current\_form = check\_normal\_form(csv\_filePath, FD, Key, MVD)

Convert to 3NF

tables = convert\_to\_3NF(FD, Key, {})