Documentation

Team Members

Ganesh Guddanti
Doondi Ashlesh Tammineedi

Inputs:

1. Database Table (.csv) file path.
2. Functional Dependencies
3. Multi Valued Dependencies
4. Key
5. Checking if user wants to find the highest normal form of Input Database Table.
6. Choice of the highest normal form to reach (1: 1NF, 2: 2NF, 3: 3NF, B: BCNF, 4: 4NF, 5: 5NF).

Output:

1. If the input table is not in 1NF then the table after converting to 1NF would be written to Output.txt file and to converted1NF.csv file
2. Output.txt file containing the SQL queries for the tables decomposed to highest normal form taken as input.
3. Output.txt file containing the details of Highest normal form of the given table if the user asks for it.

Assumptions:

Based on my input:

- Each class can be taught by multiple professors.
- Each professor can teach multiple courses.
- A class can have multiple sections i.e., multiple classroom.

- A student can take multiple courses but not the same course more than once.
- A single course with multiple sections has same start and end date.

Functional Dependencies:

- StudentID -> FirstName, LastName
- Course, Professor -> classroom
- Course -> CourseStart, CourseEnd
- Professor -> ProfessorEmail

Multi-Valued Dependencies:

- Course ->> Professor
- Course ->> classroom
- StudentID ->> Course
- StudentID ->> Professor

Keys:

- StudentID,Course

Process of Execution:

Initially inputs are taken from the user.

1. Taking the Database Table file path.
    - The .csv file path is taken as input from the user on which the normalization operations would be performed and stored in csv_filePath variable.
2. The Functional Dependencies are taken as Input.
    - A variable FD of type list is created, and all the input functional dependencies are appended to it.
    - A functional dependency can be of type A->B or A->B,C or A,B->C.
    - The variable takes input till user enters “Done” denoting that the list of functional dependencies is now completed.

3. The Multi Valued Dependencies are taken as Input.
    - A variable MVD of type list is created, and all the input multi valued dependencies are appended to it.
    - A functional dependency is of type A->>B.
    - The variable takes input till user enters “Done” denoting that the list of multi valued dependencies is now completed.
4. The Key is taken as Input.
    - Key can be primary key or composite primary key.
    - Each attribute in the key is to be separated by a comma - “,”.
5. An integer value 1 or 2 is taken as input for yes or no respectively based on user’s choice to find the highest normal form of the input table.
   If 1, then a function check_normal_form() is called as input and it’s output is written to “Output.txt” file.
6. The Highest Normal Form number is taken as input and the input table would be converted to the specified normal form.

Output :

Let us consider the user provided the path for the input and also gave all the functional dependencies and multivalued dependencies. If the user selects the integer value 1 to check whether in which normal form the table is present that output is printed to the Output.txt file and the user gave 4 as the highest normal form to check and that output also printed to the output.txt file.

CREATE TABLE StudentID (StudentID INT, FirstName VARCHAR(100) NOT NULL, LastName VARCHAR(100) NOT NULL, PRIMARY KEY (StudentID));

CREATE TABLE Course (CourseEnd DATE NOT NULL, Course VARCHAR(100), CourseStart DATE NOT NULL, PRIMARY KEY (Course));

CREATE TABLE Candidate (StudentID INT, Course VARCHAR(100), Professor VARCHAR(100), PRIMARY KEY (StudentID, Course), FOREIGN KEY (StudentID) REFERENCES StudentID(StudentID), FOREIGN KEY (Course) REFERENCES Course(Course), FOREIGN KEY (Professor) REFERENCES Professor(Professor));

CREATE TABLE Table_0 (ClassRoom VARCHAR(100) NOT NULL, Professor VARCHAR(100), Course VARCHAR(100), FOREIGN KEY (Professor) REFERENCES Professor(Professor), FOREIGN KEY (Course) REFERENCES Course(Course));

CREATE TABLE Professor (Professor VARCHAR(100), ProfessorEmail VARCHAR(50) NOT NULL, PRIMARY KEY (Professor));

The given input table is In 1NF

Logical breakdown:

1NF:

- The table already satisfies 1NF.

2NF: Removing partial dependencies

- StudentID -> FirstName, LastName
- Course, Professor -> classroom
- Course -> CourseStart, CourseEnd
- The given composite key is (StudentID, Course)
- Since Course-> CourseStart, CourseEnd, CourseStart, CourseEnd are dependent on Course. So, partial dependencies exist.
- FirstName, LastName depend only on StudentID. So, partial dependencies exist.
- So, the input table is not in 2NF and it was in 1NF.
- Breaking the partial dependencies gives the tables Student(StudentID (Primary Key), FirstName, LastName), CourseDetails(Course (Primary Key), CourseStart, CourseEnd), Enrollment(StudentID (Composite Key), Course (Composite Key), Professor, ProfessorEmail , classRoom)

3NF: Removing Transitive Dependencies

- Transitive dependencies are there only in table Enrollment as Professor -> ProfessorEmail transitively depends on the primary key.
- So, the new tables will be
- Student(StudentID (Primary Key), FirstName, LastName),

- CourseDetails(Course (Primary Key), CourseStart, CourseEnd),
- Enrollment(StudentID (Composite Key), Course (Composite Key), Professor, classRoom)
- ProfessorDetails(Professor(Primary Key), ProfessorEmail)

BCNF: Eliminating the anamolies

- Student(StudentID (Primary Key), FirstName, LastName),
- CourseDetails(Course (Primary Key), CourseStart, CourseEnd),
- Enrollment(StudentID (Composite Key), Course (Composite Key), Professor, classRoom)
    - Course, Professor -> classroom
    - StudentID, Course -> Professor, classRoom
- ProfessorDetails(Professor(Primary Key), ProfessorEmail)
- The only BCNF violating functional dependency exists in Enrollment table only, since the left hand side is not a super key (Course, Professor -> classroom).
- Updated tables:
    - Student(StudentID (Primary Key), FirstName, LastName),
    - CourseDetails(Course (Primary Key), CourseStart, CourseEnd),
    - Enrollment(StudentID (Composite Key), Course (Composite Key), Professor)
    - ClassRoomDetails(Course (Composite Key), Professor (Composite Key), classRoom)
    - ProfessorDetails(Professor(Primary Key), ProfessorEmail)

4NF: Removing Multivalued Dependencies

- Course ->> Professor
- Course ->> classroom
- StudentID ->> Course
- StudentID ->> Professor
- It removes the multivalued dependencies and the SQl queries are printed to output.txt file

Output for 5NF:

CREATE TABLE StudentID (LastName VARCHAR(100) NOT NULL, StudentID INT, FirstName VARCHAR(100) NOT NULL, PRIMARY KEY (StudentID));

CREATE TABLE Course (CourseStart DATE NOT NULL, CourseEnd DATE NOT NULL, Course VARCHAR(100), PRIMARY KEY (Course));

CREATE TABLE Candidate (StudentID INT, PRIMARY KEY (StudentID, Course), FOREIGN KEY (StudentID) REFERENCES StudentID(StudentID));

CREATE TABLE Table_0 (Course VARCHAR(100), FOREIGN KEY (Course) REFERENCES Course(Course));

CREATE TABLE Professor (Professor VARCHAR(100), ProfessorEmail VARCHAR(50) NOT NULL, PRIMARY KEY (Professor));

CREATE TABLE Decomposed_5 (Professor VARCHAR(100), Course VARCHAR(100), FOREIGN KEY (Professor) REFERENCES Professor(Professor), FOREIGN KEY (Course) REFERENCES Course(Course));

CREATE TABLE Decomposed_7 (Course VARCHAR(100), ClassRoom VARCHAR(100) NOT NULL, FOREIGN KEY (Course) REFERENCES Course(Course));

CREATE TABLE Decomposed_8 (Course VARCHAR(100), StudentID INT, FOREIGN KEY (Course) REFERENCES Course(Course), FOREIGN KEY (StudentID) REFERENCES StudentID(StudentID));

The given input table is In 1NF