DBMS project to create Functional dependency on data fields using MySQL and Python

Project by Ayyappa Kirla

In [2]:
```
# Import packages
import sqlite3
from sqlalchemy import create_engine
from tabulate import tabulate

class Functional_Dependency:
        def __init__(self):
                """
                Create the database FD_DB and tables Instructors, Students and FD.
                """

                con = sqlite3.connect('FD_DB.db')
                cur = con.cursor()

                cur.execute('''CREATE TABLE if not exists Instructors
                                        (Instructor text, Department text)''')

                cur.execute('''CREATE TABLE if not exists Students
                                        (Roll_Number text PRIMARY KEY, First_Name text, Surname text,Department text)''')

                cur.execute('''CREATE TABLE if not exists FD
                                        (Table_Name text, Determinant text, Dependant text)''')

                con.commit()
                con.close()


        def insert_Instructors(self,instructors):
                """
                Insert the given list of instructors into the Instructor table
                Arguments:
                instructors : list of lists
                Returns:
```

```
            Nothing
            """

            con = sqlite3.connect('FD_DB.db')
            cur = con.cursor()

            try:
                    cur.executemany("INSERT INTO Instructors VALUES(?, ?)", instructors)
                    con.commit()
                    con.close()
            except sqlite3.Error as er:
                    print('SQLite error: %s' % (' '.join(er.args)))
                    print("Exception class is: ", er.__class__)
                    print('Error occured while inserting into Instructors')


    def insert_Students(self,students):
            """
            Insert the given list of students into the Student table
            Arguments:
            students : list of lists
            Returns:
            Nothing
            """

            con = sqlite3.connect('FD_DB.db')
            cur = con.cursor()

            try:
                    cur.executemany("INSERT INTO Students VALUES(?, ?, ?, ?)", students)
                    con.commit()
                    con.close()
            except sqlite3.Error as er:
                    print('SQLite error: %s' % (' '.join(er.args)))
                    print("Exception class is: ", er.__class__)
                    print('Error occured while inserting into Students')

    def print_data(self,table_name):
            """
            Print the data of table name if provided, else print all tables.
            Arguments:
            table_name : Name of the table to be printed
            Returns:
            Nothing
```

```
    """

    con = sqlite3.connect('FD_DB.db')
    cur = con.cursor()

    if table_name == 'Instructors':

            data = cur.execute('''SELECT  Instructor, Department FROM Instructors''')
            headers = ['Instructor', 'Department']

            print(tabulate(data,headers,tablefmt="fancy_grid"))

    elif table_name == 'Students':

            data = cur.execute('''SELECT  Roll_Number, First_Name, Surname, Department FROM Students''')
            headers = ['Roll_Number', 'First_Name', 'Surname', 'Department']

            print(tabulate(data,headers,tablefmt="fancy_grid"))

    elif table_name == 'FD':


            data = cur.execute('''SELECT  Table_Name, Determinant, Dependant FROM FD''')
            headers = ['Table_Name', 'Determinant', 'Dependant']

            print(tabulate(data,headers,tablefmt="fancy_grid"))

    else:

            data = cur.execute('''SELECT  Instructor, Department FROM Instructors''')
            headers = ['Instructor', 'Department']

            print(tabulate(data,headers,tablefmt="fancy_grid"))

            data = cur.execute('''SELECT  Roll_Number, First_Name, Surname, Department FROM Students''')
            headers = ['Roll_Number', 'First_Name', 'Surname', 'Department']

            print(tabulate(data,headers,tablefmt="fancy_grid"))

            data = cur.execute('''SELECT  Table_Name, Determinant, Dependant FROM FD''')
            headers = ['Table_Name', 'Determinant', 'Dependant']

            print(tabulate(data,headers,tablefmt="fancy_grid"))
```

```
                con.commit()
                con.close()


        def delete_data(self):
                """
                This method will delete all the records of all three tables.
                Arguments:
                Nothing

                Returns:
                Nothing
                """

                con = sqlite3.connect('FD_DB.db')
                cur = con.cursor()

                cur.execute("DELETE FROM Instructors")
                cur.execute("DELETE FROM Students")
                cur.execute("DELETE FROM FD")

                con.commit()
                con.close()

        def check_status_functional_dependency(self, table_name, determinant , dependant):
                """
                This method checks if a functional dependency still hold on the table
                Arguments:
                table_name : Table on which FD is to be checked
                determinant : The set of determining attributes
                dependant : The dependant attribute
                Returns:
                0 if violation is found
                1 if no violation is found
                """
                conn = sqlite3.connect('FD_DB.db')
                conn.row_factory = sqlite3.Row
                c = conn.cursor()
                c.execute("SELECT {},COUNT(DISTINCT {}) as count from {} group by {}".format(determinant,
                        dependant,table_name,determinant))
                result = [dict(row) for row in c.fetchall()]
                listy = []
                for item in result:
                        listy.append(item['count'])
```

```
                max_value = max(listy)

                if max_value > 1:
                        return 0
                else:
                        return 1

        def create_functional_dependency(self, table_name, determinant , dependant):
                """
                Create a functional dependency on a table
                Arguments:
                table_name : Table on which FD is to be created
                determinant : The set of determining attributes
                dependant : The dependant attribute
                Returns:
                Nothing
                """

                status = self.check_status_functional_dependency(table_name,determinant,dependant)
                if status == 1:

                        con = sqlite3.connect('FD_DB.db')
                        cur = con.cursor()
                        index_name = table_name + determinant + dependant

                        try:
                                cur.execute("CREATE INDEX {} on {} ({}, {})".format(index_name,table_name,determinant,dependant))
                                con.commit()
                                print('Index created Successfully on Table ',table_name,' for ',determinant,' and ',dependant)
                        except sqlite3.Error as er:
                                print('SQLite error: %s' % (' '.join(er.args)))
                                print("Exception class is: ", er.__class__)
                                print('Error occured while creating INDEX')
                                pass

                        fd_row = [table_name,determinant,dependant]

                        try:
                                cur.execute("INSERT INTO FD VALUES(?, ?, ?)", fd_row)
                                con.commit()
                                con.close()
                        except sqlite3.Error as er:
                                print('SQLite error: %s' % (' '.join(er.args)))
                                print("Exception class is: ", er.__class__)
```

```
                                print('Error occured while inserting into FD table')

                        print('Functional Dependency created Successfully on Table ',table_name,
                                ' for ',determinant,' and ',dependant)
                        self.print_data(table_name)

                else:
                        print('Fuctional Dependency is not created on Table ',table_name,
                                ' for ',determinant,' and ',dependant ,' because of the existing data inside the table')
                        self.print_data(table_name)


        def query_executor(self,query):
                """
                Execute the query
                Arguments:
                query : Query
                Returns:
                Nothing
                """

                con = sqlite3.connect('FD_DB.db')
                cur = con.cursor()

                try:
                        cur.execute(query)
                        con.commit()
                        con.close()
                        print('Query executed Successfully')

                except sqlite3.Error as er:
                        print('SQLite error: %s' % (' '.join(er.args)))
                        print("Exception class is: ", er.__class__)
                        print('Error occured while executing the query',query)

                conn = sqlite3.connect('FD_DB.db')
                conn.row_factory = sqlite3.Row
                c = conn.cursor()
                c.execute('''SELECT  Table_Name, Determinant, Dependant FROM FD''')

                result = [dict(row) for row in c.fetchall()]

                for item in result:
                        status = self.check_status_functional_dependency(item['Table_Name'],item['Determinant'],item['Dependant'
```

```
                        if status == 1:
                                print('Functional Dependency on ',item['Table_Name'],' for ',item['Determinant'],' and ',
                                        item['Dependant'],' is not violated after the execution of query ', query)
                                self.print_data(item['Table_Name'])
                        else:
                                print('Functional Dependency on ',item['Table_Name'],' for ',item['Determinant'],' and ',
                                        item['Dependant'], ' is violated after the execution of query', query)
                                self.print_data(item['Table_Name'])

if __name__ == "__main__":


        test = Functional_Dependency()

        instructors = [['Sudarshan','MEMS'],['CV Raman','Physics'],['Sudarshan','EE'],['Ramanujam','Mathematics']]
        students = [['MM601','Ayyappa','Kirla','MEMS'],['MM602','Nani','Iyer','EE'],
                ['MM603','Mahendra Singh','Dhoni','MEMS']]


        print('INITIAL STATUS OF THE DB')
        test.delete_data()
        test.insert_Instructors(instructors)
        test.insert_Students(students)
        test.print_data('ALL')

        print('-----------------------------------------------------------------------------------------')

        print('Creating FD on Instructors table')
        test.create_functional_dependency('Instructors', 'Instructor' , 'Department')

        print('Creating FD on Students table for Roll_Number and First_Name')
        test.create_functional_dependency('Students', 'Roll_Number' , 'First_Name')

        print('Creating FD on Students table for First_Name, Surname and Department')
        test.create_functional_dependency('Students', "First_Name, Surname" , 'Department')

        print('-----------------------------------------------------------------------------------------')

        test.print_data('FD')

        print('Inserting and Updating DB')
        print('Inserting will check ONLY tables in the FD table, for FD violation')

        insert_query_1 = "INSERT INTO Instructors VALUES('Marvel', 'Science Fiction')"
```

```
        test.query_executor(insert_query_1)

        insert_query_2 = "INSERT INTO Students VALUES('MM605', 'Virat', 'Kohli', 'MEMS')"
        test.query_executor(insert_query_2)

        insert_query_3 = "INSERT INTO Students VALUES('MM606', 'Ayyappa', 'abc', 'EE')"
        test.query_executor(insert_query_3)

        insert_query_4 = "INSERT INTO Students VALUES('EE607', 'abc', 'Kirla', 'EE')"
        test.query_executor(insert_query_4)

        insert_query_5 = "INSERT INTO Students VALUES('EE608', 'Ayyappa', 'Kirla', 'EE')"
        test.query_executor(insert_query_5)

        update_query_1 = "UPDATE Students SET First_Name='Ayyappa' WHERE First_Name='Nani'"
        test.query_executor(update_query_1)
```

INITIAL STATUS OF THE DB

Instructor	Department
Sudarshan	MEMS
CV Raman	Physics
Sudarshan	EE
Ramanujam	Mathematics

Roll_Number	First_Name	Surname	Department
MM601	Ayyappa	Kirla	MEMS
MM602	Nani	Iyer	EE
MM603	Mahendra Singh	Dhoni	MEMS

Table_Name	Determinant	Dependant

Creating FD on Instructors table
Fuctional Dependency is not created on Table Instructors for Instructor and Department because of the existing data inside the table

Instructor	Department
Sudarshan	MEMS
CV Raman	Physics
Sudarshan	EE
Ramanujam	Mathematics

Creating FD on Students table for Roll_Number and First_Name
SQLite error: index StudentsRoll_NumberFirst_Name already exists
Exception class is: <class 'sqlite3.OperationalError'>
Error occured while creating INDEX

Functional Dependency created Successfully on Table Students for Roll_Number and First_Name

Roll_Number	First_Name	Surname	Department
MM601	Ayyappa	Kirla	MEMS
MM602	Nani	Iyer	EE
MM603	Mahendra Singh	Dhoni	MEMS

Creating FD on Students table for First_Name, Surname and Department
SQLite error: near ",": syntax error
Exception class is: <class 'sqlite3.OperationalError'>
Error occured while creating INDEX
Functional Dependency created Successfully on Table Students for First_Name, Surname and Department

Roll_Number	First_Name	Surname	Department
MM601	Ayyappa	Kirla	MEMS
MM602	Nani	Iyer	EE
MM603	Mahendra Singh	Dhoni	MEMS

Table_Name	Determinant	Dependant
Students	Roll_Number	First_Name
Students	First_Name, Surname	Department

Inserting and Updating DB
Inserting will check ONLY tables in the FD table, for FD violation
Query executed Successfully
Functional Dependency on Students for Roll_Number and First_Name is not violated after the execution of query INSERT INTO Instructors VALUES('Marvel', 'Science Fiction')

Roll_Number	First_Name	Surname	Department
MM601	Ayyappa	Kirla	MEMS
MM602	Nani	Iyer	EE

MM603	Mahendra Singh	Dhoni	MEMS

Functional Dependency on Students for First_Name, Surname and Department is not violated after the execution of query INSERT INTO Instructors VALUES('Marvel', 'Science Fiction')

Roll_Number	First_Name	Surname	Department
MM601	Ayyappa	Kirla	MEMS
MM602	Nani	Iyer	EE
MM603	Mahendra Singh	Dhoni	MEMS

Query executed Successfully
Functional Dependency on Students for Roll_Number and First_Name is not violated after the execution of query INSERT INTO Students VALUES('MM605', 'Virat', 'Kohli', 'MEMS')

Roll_Number	First_Name	Surname	Department
MM601	Ayyappa	Kirla	MEMS
MM602	Nani	Iyer	EE
MM603	Mahendra Singh	Dhoni	MEMS
MM605	Virat	Kohli	MEMS

Functional Dependency on Students for First_Name, Surname and Department is not violated after the execution of query INSERT INTO Students VALUES('MM605', 'Virat', 'Kohli', 'MEMS')

Roll_Number	First_Name	Surname	Department
MM601	Ayyappa	Kirla	MEMS
MM602	Nani	Iyer	EE
MM603	Mahendra Singh	Dhoni	MEMS
MM605	Virat	Kohli	MEMS

Query executed Successfully
Functional Dependency on Students for Roll_Number and First_Name is not violated after the execution of query INSERT INTO Students VALUES('MM606', 'Ayyappa', 'abc', 'EE')

Roll_Number	First_Name	Surname	Department
MM601	Ayyappa	Kirla	MEMS
MM602	Nani	Iyer	EE
MM603	Mahendra Singh	Dhoni	MEMS
MM605	Virat	Kohli	MEMS
MM606	Ayyappa	abc	EE

Functional Dependency on Students for First_Name, Surname and Department is not violated after the execution of query INSERT INTO Students VALUES('MM606', 'Ayyappa', 'abc', 'EE')

Roll_Number	First_Name	Surname	Department
MM601	Ayyappa	Kirla	MEMS
MM602	Nani	Iyer	EE
MM603	Mahendra Singh	Dhoni	MEMS
MM605	Virat	Kohli	MEMS
MM606	Ayyappa	abc	EE

Query executed Successfully
Functional Dependency on Students for Roll_Number and First_Name is not violated after the execution of query INSERT INTO Students VALUES('EE607', 'abc', 'Kirla', 'EE')

Roll_Number	First_Name	Surname	Department
MM601	Ayyappa	Kirla	MEMS
MM602	Nani	Iyer	EE
MM603	Mahendra Singh	Dhoni	MEMS
MM605	Virat	Kohli	MEMS
MM606	Ayyappa	abc	EE
EE607	abc	Kirla	EE

Functional Dependency on Students for First_Name, Surname and Department is not violated after the execution of query INSERT INTO Students VALUES('EE607', 'abc', 'Kirla', 'EE')

Roll_Number	First_Name	Surname	Department
MM601	Ayyappa	Kirla	MEMS
MM602	Nani	Iyer	EE
MM603	Mahendra Singh	Dhoni	MEMS
MM605	Virat	Kohli	MEMS
MM606	Ayyappa	abc	EE
EE607	abc	Kirla	EE

Query executed Successfully

Functional Dependency on Students for Roll_Number and First_Name is not violated after the execution of query INSERT INTO Students VALUES('EE608', 'Ayyappa', 'Kirla', 'EE')

Roll_Number	First_Name	Surname	Department
MM601	Ayyappa	Kirla	MEMS
MM602	Nani	Iyer	EE
MM603	Mahendra Singh	Dhoni	MEMS
MM605	Virat	Kohli	MEMS
MM606	Ayyappa	abc	EE
EE607	abc	Kirla	EE
EE608	Ayyappa	Kirla	EE

Functional Dependency on Students for First_Name, Surname and Department is violated after the execution of query INSERT INTO Students VALUES('EE608', 'Ayyappa', 'Kirla', 'EE')

Roll_Number	First_Name	Surname	Department
MM601	Ayyappa	Kirla	MEMS

MM602	Nani	Iyer	EE
MM603	Mahendra Singh	Dhoni	MEMS
MM605	Virat	Kohli	MEMS
MM606	Ayyappa	abc	EE
EE607	abc	Kirla	EE
EE608	Ayyappa	Kirla	EE

Query executed Successfully
Functional Dependency on Students for Roll_Number and First_Name is not violated after the execution of query UPDATE Students SET First_Name='Ayyappa' WHERE First_Name='Nani'

Roll_Number	First_Name	Surname	Department
MM601	Ayyappa	Kirla	MEMS
MM602	Ayyappa	Iyer	EE
MM603	Mahendra Singh	Dhoni	MEMS
MM605	Virat	Kohli	MEMS
MM606	Ayyappa	abc	EE
EE607	abc	Kirla	EE
EE608	Ayyappa	Kirla	EE

Functional Dependency on Students for First_Name, Surname and Department is violated after the execution of query UPDATE Students SET First_Name='Ayyappa' WHERE First_Name='Nani'

Roll_Number	First_Name	Surname	Department
MM601	Ayyappa	Kirla	MEMS
MM602	Ayyappa	Iyer	EE
MM603	Mahendra Singh	Dhoni	MEMS

MM605	Virat	Kohli	MEMS
MM606	Ayyappa	abc	EE
EE607	abc	Kirla	EE
EE608	Ayyappa	Kirla	EE

In [ ]: