LAPORAN

RENCANA TUGAS MAHASISWA Ke-2

MATA KULIAH ALGORITMA DAN PEMROGRAMAN LANJUT

"Merancang class diagram untuk kebutuhan desain sistem basis data dan mengimplementasikan ke dalam bahasa Pemrograman Python CRUD OOP terintegrasi ke basis data MySQL"

DISUSUN OLEH:

Muhammad Aryasatya Nugroho (NPM 22083010085)

DOSEN PENGAMPU:

Tresna Maulana Fahrudin S.ST., M.T. (NIP. 199305012022031007)

PROGRAM STUDI SAINS DATA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN" JAWA TIMUR

2023

Soal : Implementasikan Pembuataan Class CRUD Database pada rancangan Class Diagram yang pernah Anda buat sebelumnya

Class Diagram:

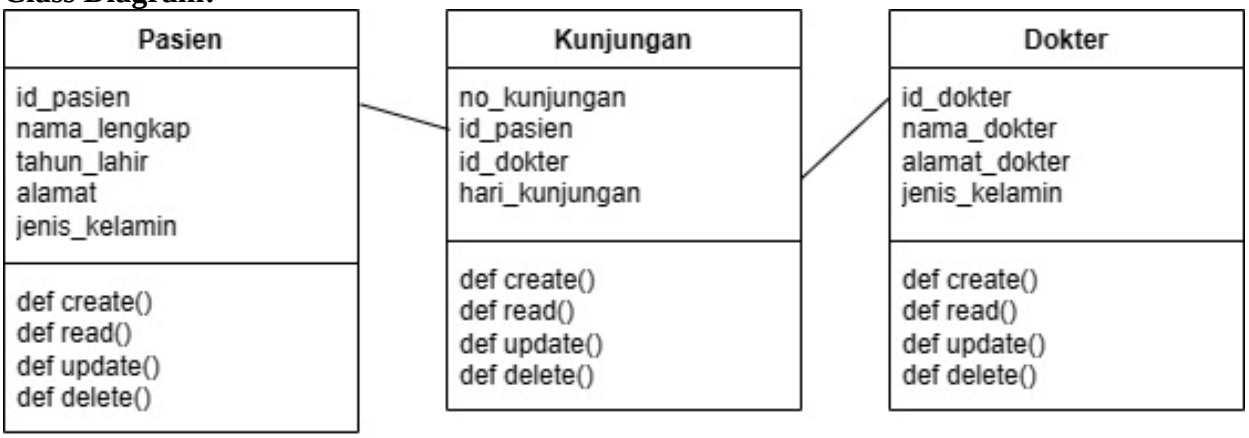
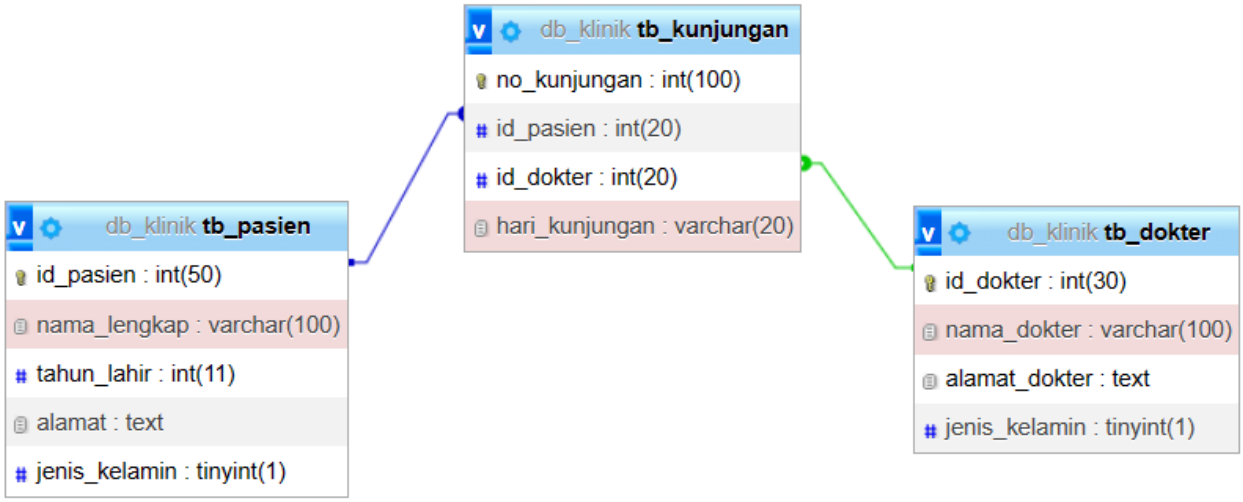


Diagram UML diatas memiliki 3 kelas yaitu:

1) Kelas Dokter: kelas ini memiliki atribut id dokter, nama, alamat dokter, dan jenis kelamin.
2) Kelas Pasien: Kelas ini akan memiliki atribut seperti id pasien, nama lengkap, tahun lahir, alamat, dan jenis kelamin.
3) Kelas kunjungan: Kelas ini mempresentasikan setiap kunjungan yang dilakukan pasien ke dokter, maka kelas ini memiliki atribut seperti nomor kunjungan, id pasien, id dokter, dan hari kunjungan.
4)

Visualisasi Relasi:



Kode Script:

```
In [76]:
import mysql.connector

my_db = mysql.connector.connect(
            host="localhost",
            user="root",
            password="",
            database="db_klinik"
        )
dp = my_db.cursor()

if my_db.is_connected():
    print("OKEEE")
```

```
OKEEE
```

Pada kode diatas, dibuat koneksi ke server MySQL dengan mengisi detail host, username, password, dan nama database pada parameter fungsi connect(). Pastikan Modul MySql Connector Python terpasang dan database MySql sudah di install dan diaktifkan pada perangkat anda sebelum menjalankan kode ini.
Setelah melakukan koneksi ke server, kita perlu membuat objek cursor untuk melakukan operasi pada database. Kemudian, kita dapat mengecek apakah koneksi berhasil dilakukan dengan menggunakan metode is_connected(). Jika berhasil connect, maka pesan "OKE" akan ditampilkan.

Kelas Pasien:

```
In [77]:
class tb_pasien:
    def add_tb_pasien(self, nama_lengkap, tahun_lahir, alamat, jenis_kelamin):
        try:
            sql = "INSERT INTO tb_pasien (nama_lengkap, tahun_lahir, alamat, jenis_kelamin) VALUES (%s,
            val = (nama_lengkap, tahun_lahir, alamat, jenis_kelamin)
            dp.execute(sql, val)
            my_db.commit()
            print("Berhasil ditambahkan tb_pasien")
        except Exception as e:
            print("Error: ", e)

    def read_tb_pasien(self):
        try:
            sql = "SELECT * FROM tb_pasien"
            dp.execute(sql)
            result = dp.fetchall()
            for row in result:
                print(row)
        except Exception as e:
            print("Error: ", e)
```

```python
def update_tb_pasien(self, id_pasien, nama_lengkap, tahun_lahir, alamat, jenis_kelamin):
    try:
        sql = "UPDATE tb_pasien SET nama_lengkap=%s, tanggal_lahir=%s, alamat=%s, jenis_kelamin=%s
        val = (nama_lengkap, tahun_lahir, alamat, jenis_kelamin, id_pasien)
        dp.execute(sql, val)
        my_db.commit()
        print("Berhasil diupdate tb_pasien")
    except Exception as e:
        print("Error: ", e)

def delete_tb_pasien(self, id_pasien):
    try:
        sql = "DELETE FROM tb_pasien WHERE id_pasien=%s"
        val = (id_pasien,)
        dp.execute(sql, val)
        my_db.commit()
        print("Berhasil dihapus tb_pasien")
    except Exception as e:
        print("Error: ", e)
```

Kode di atas merupakan implementasi dari sebuah Class tb_pasien, digunakan untuk melakukan operasi CRUD (Create, Read, Update, Delete) pada tabel tb_pasien di database. Class pasien memiliki empat method dibawah ini:

1) add_tb_pasien(): Metode ini digunakan untuk menambahkan data pasien baru ke tabel tb_pasien. Metode ini membutuhkan 4 parameter yaitu nama_lengkap, tahun_lahir, alamat, dan jenis kelamin. Parameter-parameter ini digunakan untuk memasukkan nilai-nilai baru ke dalam kolom yang sesuai pada tabel tb_pasien.
2) read_tb_pasien(): Method ini digunakan untuk membaca seluruh isi tabel tb_pasien dan menampilkannya ke layar. Method ini tidak membutuhkan parameter.
3) update_tb_pasien(): Method ini digunakan untuk memperbarui data pasien yang sudah ada di tabel tb_pasien. Method ini membutuhkan lima parameter yaitu id_pasien, nama_lengkap, tahun_lahir, alamat, dan jenis_kelamin. Parameter id_pasien digunakan untuk memilih baris mana yang akan diperbarui, sedangkan parameter-parameter lainnya digunakan untuk memperbarui nilai-nilai di kolom-kolom yang sesuai pada tabel tb_pasien.
4) delete_tb_pasien(): Metode ini Digunakan untuk menghapus data pasien dari tabel "tb_pasien". Metode ini menerima satu parameter yaitu "id_pasien", yang akan digunakan sebagai kondisi untuk menghapus data pasien dari tabel.

Kelas Kunjungan:

```
In [137]:
class tb_kunjungan:
    def add_tb_kunjungan(self, no_kunjungan, id_pasien, id_dokter, hari_kunjungan):
        try:
            sql = "INSERT INTO tb_kunjungan (no_kunjungan, id_pasien, id_dokter, hari_kunjungan) VALUES
            val = (no_kunjungan, id_pasien, id_dokter, hari_kunjungan)
            dp.execute(sql, val)
            my_db.commit()
        except Exception as e:
            print("Error: ",e)

    def read_tb_kunjungan(self, no_kunjungan):
        try:
            sql = "SELECT * FROM tb_kunjungan WHERE no_kunjungan = %s"
            val = (no_kunjungan,)
            dp.execute(sql, val)
            result = dp.fetchall()
            return result
        except Exception as e:
            print("Error: ", e)

    def update_tb_kunjungan(self, id_dokter, no_kunjungan):
        try:
            sql = "UPDATE tb_kunjungan SET id_dokter = %s WHERE no_kunjungan = %s"
            val = (id_dokter, no_kunjungan)
            dp.execute(sql, val)
            my_db.commit()
            print("Data tb_kunjungan berhasil diupdate")
        except Exception as e:
            print("Error: ", e)

    def delete_tb_kunjungan(self, no_kunjungan):
        try:
            sql = "DELETE FROM tb_kunjungan WHERE no_kunjungan = %s"
            val = (no_kunjungan,)
            dp.execute(sql, val)
            my_db.commit()
            print("Data tb_kunjungan berhasil dihapus")
        except Exception as e:
            print("Error: ", e)
```

Kode di atas merupakan implementasi dari sebuah Class tb_kunjungan, digunakan untuk melakukan operasi CRUD (Create, Read, Update, Delete) pada tabel tb_kunjungan di database. Class kunjungan memiliki empat method dibawah ini:

1) add_tb_kunjungan: Metode ini dapat menambahkan data ke tabel tb_kunjungan. Memiliki empat parameter yaitu no_kunjungan, id_pasien, id_dokter, dan hari_kunjungan. Metode ini akan melakukan eksekusi SQL untuk menambahkan data ke dalam tabel tb_kunjungan.

2) read_tb_kunjungan: Metode ini digunakan untuk membaca data dari tabel tb_kunjungan. Metode ini menerima satu parameter, yaitu no_kunjungan. Metode ini akan melakukan eksekusi SQL untuk membaca data dari tabel tb_kunjungan berdasarkan nomor kunjungan yang diberikan. Metode ini akan mengembalikan hasil dari eksekusi SQL dalam bentuk tuple.
3) update_tb_kunjungan: Metode ini digunakan untuk mengupdate data pada tabel tb_kunjungan. Metode mempunyai dua parameter, yaitu id_dokter dan no_kunjungan. Metode ini akan melakukan eksekusi SQL untuk mengupdate data pada tabel tb_kunjungan berdasarkan nomor kunjungan yang diberikan.
4) delete_tb_kunjungan: Metode ini digunakan untuk menghapus data pada tabel tb_kunjungan. Metode ini menerima satu parameter, yaitu no_kunjungan saja. Metode ini akan melakukan eksekusi SQL untuk menghapus data pada tabel tb_kunjungan berdasarkan nomor kunjungan yang diberikan.

Kelas Dokter:

```
In [138]:
class tb_dokter:
    def add_tb_dokter(self, id_dokter, nama_dokter, alamat_dokter, jenis_kelamin):
        try:
            sql = "INSERT INTO tb_dokter (id_dokter, nama_dokter, alamat_dokter, jenis_kelamin) VALUES
            val = (id_dokter, nama_dokter, alamat_dokter, jenis_kelamin)
            dp.execute(sql, val)
            my_db.commit()
        except Exception as e:
            print("Error: ",e)

    def read_tb_dokter(self, id_dokter):
        try:
            sql = "SELECT * FROM tb_dokter WHERE id_dokter = %s"
            val = (id_dokter,)
            dp.execute(sql, val)
            result = dp.fetchall()
            return result
        except Exception as e:
            print("Error: ", e)
def update_tb_dokter(self, id_dokter, alamat_dokter):
    try:
        sql = "UPDATE tb_dokter SET alamat_dokter = %s WHERE id_dokter = %s"
        val = (alamat_dokter, id_dokter)
        dp.execute(sql, val)
        my_db.commit()
    except Exception as e:
        print("Error: ", e)

def delete_tb_dokter(self, id_dokter):
    try:
        sql = "DELETE FROM tb_dokter WHERE id_dokter = %s"
        val = (id_dokter,)
        dp.execute(sql, val)
        my_db.commit()
    except Exception as e:
        print("Error: ", e)
```

Kode di atas merupakan implementasi dari sebuah Class tb_dokter, digunakan untuk melakukan operasi CRUD (Create, Read, Update, Delete) pada tabel tb_dokter di database. Class dokter memiliki empat method dibawah ini:

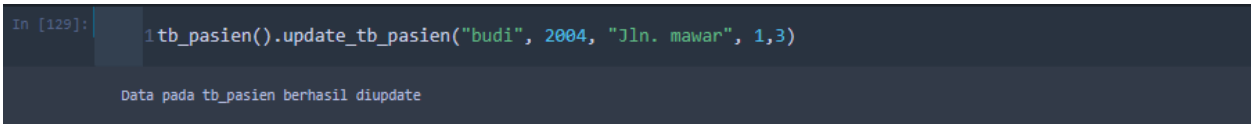
1) add_tb_dokter: Metode ini memiliki fugnsi untuk menambahkan data dokter baru ke dalam tabel tb_dokter. Memiliki empat parameter yang terdiri dari id_dokter, nama_dokter, alamat_dokter, dan jenis_kelamin, sebagai data dokter yang akan ditambahkan ke dalam tabel. Menggunakan SQL statement INSERT INTO untuk memasukkan data baru ke dalam tabel dan Jika terdapat error pada eksekusi SQL statement, maka akan di-handle dengan blok try-except dan menampilkan pesan error tersebut.
2) read_tb_dokter: Metode ini digunakan untuk membaca data dokter dari tabel tb_dokter berdasarkan id_dokter. Menerima satu parameter saja id_dokter sebagai pencarian data dokter berdasarkan id_dokter.
3) update_tb_dokter: Metode ini digunakan untuk mengupdate data pada tabel tb_kunjungan. Metode mempunyai dua parameter, yaitu id_dokter dan alamat_dokter sebagai data yang ingin diubah pada tabel dokter.
4) delete_tb_dokter: Metode ini digunakan untuk menghapus data dokter pada tabel tb_dokter berdasarkan id_dokter. Menerima satu parameter id_dokter sebagai data dokter yang akan dihapus dari tabel.

Hasil:

- **Create tb_pasien**

- **Read tb_pasien**

- **Update tb_pasien**



	id_pasien	nama_lengkap	tahun_lahir	alamat	jenis_kelamin
Edit Copy Delete	3	budi	2004	Jln. mawar	1
Edit Copy Delete	4	taufik	2003	jalan jalan	1

Check all With selected: Edit Copy Delete Export

- **Delete tb_pasien**

```
In [148]: 1 tb_pasien.delete_tb_pasien("3", "id_pasien_value")

          Data pada tb_pasien berhasil dihapus
```

	id_pasien	nama_lengkap	tahun_lahir	alamat	jenis_kelamin
Edit Copy Delete	4	taufik	2003	jalan jalan	1

Check all With selected: Edit Copy Delete Export

- **Create and Read tb_kunjungan**

```
In [ ]: 1 tb_kunjungan().add_tb_kunjungan('4', '2', 'selasa')

In [164]: 1 tb_kunjungan().read_tb_kunjungan("2")
```

	no_kunjungan	id_pasien	id_dokter	hari_kunjungan
Edit Copy Delete	1	4	2	selasa

Check all With selected: Edit Copy Delete Export

- **Update and Delete tb_kunjungan**

```
In [177]: 1 tb_kunjungan().update_tb_kunjungan("1", "2")

          Data tb_kunjungan berhasil diupdate

In [175]: 1 kunjungan.delete_tb_kunjungan(1)
```

no_kunjungan	id_pasien	id_dokter	hari_kunjungan

Query results operations

Create view

- **Create and Read tb_dokter**

```
In [187]:
1 dokter.add_tb_dokter("D001", "Dr. John Doe", "Jl. Dokter No.1", "Laki-Laki")
```

```
In [190]:
1 result = tb_dokter().read_tb_dokter(4)
```

	id_dokter	nama_dokter	alamat_dokter	jenis_kelamin
Edit Copy Delete	4	Dr. John Doe	Jl. Dokter No.1	0

- **Update and Delete tb_dokter**

```
In [167]:
1 tb_kunjungan().update_tb_kunjungan("D002", "2")
```

```
Data tb_kunjungan berhasil diupdate
```

```
In [183]:
1 dokter.delete_tb_dokter(2)
```

```
Data tb_dokter berhasil dihapus
```