**Міністерство освіти і науки України**

**Національний університет «Львівська політехніка»**

**Кафедра систем штучного інтелекту**

**Звіт**

**До лабораторної роботи №2-7**

з дисципліни:

**«Системний аналіз»**

Варіант 24

*Виконав:*

*студент групи КН-309*

*Фарина П.Н.*

*Прийняла:*

*Бойко Н. І.*

Львів – 2019р.

**Мета:** Навчитися створювати базу даних за діаграмами, патерни для роботи з базою даних, та підключати патерни до бази даних.

Decorator

import pymysql.cursors

def connect():

connection = pymysql.connect(

host = "localhost",

user = "root",

password = "qwertyui",

db = "sys\_analis",

charset = "utf8mb4",

cursorclass=pymysql.cursors.DictCursor

)

print("connected successfully!!!")

return connection

class operand1():

def get\_consultant():

connection = connect()

try:

with connection.cursor() as cursor:

sql = "select \* from consultant"

cursor.execute(sql)

print()

ret=[]

for row in cursor:

print(row)

ret.append(row)

finally:

connection.close()

return ret

def get\_polis():

connection = connect()

try:

with connection.cursor() as cursor:

sql = "select \* from polis"

cursor.execute(sql)

print()

ret=[]

for row in cursor:

print(row)

ret.append(row)

finally:

connection.close()

return ret

class operand2():

def get\_operation():

connection = connect()

try:

with connection.cursor() as cursor:

sql = "select \* from operation"

cursor.execute(sql)

print()

ret=[]

for row in cursor:

print(row)

ret.append(row)

finally:

connection.close()

return ret

class main():

def decorator():

first = operand1.get\_consultant()

print("operand 1 is being decorated:")

print()

print("idconsultant \t name \t\t level \t structure \t commission \t\n")

for row in first:

print(row["idconsultant"], " \t ", row["name"], " \t ", row["level"], " \t ", row["structure"], " \t ", row["commission"], " \t")

print("\n")

second = operand2.get\_operation()

print("operand 2 is being decorated:")

print()

print("idoperation \t data\_start \t data\_finish \t\n")

for row in second:

print(row["idoperation"], " \t\t ", row["data\_start"], "\t", row["data\_finish"], "\t")

print("\n")

if \_\_name\_\_ == "\_\_main\_\_":

main.decorator()

Adapter

import pymysql.cursors

print\_db\_row=True

type\_test=False

def connect():

connection = pymysql.connect(

host = "localhost",

user = "root",

password = "qwertyui",

db = "sys\_analis",

charset = "utf8mb4",

cursorclass=pymysql.cursors.DictCursor

)

print("connected successfully!!!")

return connection

class printer():

def show\_beautifully(array):

for x in array:

for y in x:

if(type(y)==type("test") and len(y)<9):

print(y, "\t\t", end=" ")

elif(type(y)==type("test") and len(y)<17):

print(y, "\t\t", end=" ")

elif(type(y)==type("test") and len(y)>17):

print(y, "\t", end=" ")

elif(type(y)==type(type\_test)):

print(y, "\t\t", end=" ")

else:

print(y, "\t\t", end=" ")

print("\n", end=" ")

class operand():

def get\_consultant():

connection = connect()

try:

with connection.cursor() as cursor:

sql = "select \* from consultant"

cursor.execute(sql)

print()

ret=[]

for row in cursor:

if (print\_db\_row==True):

print(row)

ret.append(row)

finally:

connection.close()

return ret

def get\_polis():

connection = connect()

try:

with connection.cursor() as cursor:

sql = "select \* from polis"

cursor.execute(sql)

print()

ret=[]

for row in cursor:

if (print\_db\_row==True):

print(row)

ret.append(row)

finally:

connection.close()

global type\_test

type\_test=ret[1]["commission"]

return ret

class main():

def adapter(dbarray):

output=[]

a=[]

for b in dbarray[0]:

a.append(b)

output.append(a)

for row in dbarray:

elements=[]

for element in row:

elements.append(row[element])

output.append(elements)

return output

printer.show\_beautifully(main.adapter(operand.get\_consultant()))

Singelton

import pymysql.cursors

print\_db\_row = False

type\_test = False

def connect():

connection = pymysql.connect(

host="localhost",

user="root",

password="qwertyui",

db="sys\_analis",

charset="utf8mb4",

cursorclass=pymysql.cursors.DictCursor

)

print("connected successfully!!!")

return connection

class operand():

def get\_consultant():

connection = connect()

try:

with connection.cursor() as cursor:

sql = "select \* from consultant"

cursor.execute(sql)

print()

ret = []

for row in cursor:

if (print\_db\_row == True):

print(row)

ret.append(row)

finally:

connection.close()

return ret

def get\_polis():

connection = connect()

try:

with connection.cursor() as cursor:

sql = "select \* from polis"

cursor.execute(sql)

print()

ret = []

for row in cursor:

if (print\_db\_row == True):

print(row)

ret.append(row)

finally:

connection.close()

global type\_test

type\_test = ret[0]["name"]

return ret

class printer():

def show\_beautifully(array):

for x in array:

for y in x:

if(type(y)==type("test") and len(y)<9):

print(y, "\t\t", end=" ")

elif(type(y)==type("test") and len(y)<17):

print(y, "\t\t", end=" ")

elif(type(y)==type("test") and len(y)>17):

print(y, "\t", end=" ")

elif(type(y)==type(type\_test)):

print(y, "\t\t", end=" ")

else:

print(y, "\t\t", end=" ")

print("\n", end=" ")

def adapter(dbarray):

output = []

a = []

for b in dbarray[0]:

a.append(b)

output.append(a)

for row in dbarray:

elements = []

for element in row:

elements.append(row[element])

output.append(elements)

return output

class Singleton:

\_\_instance = None

@staticmethod

def getInstance():

""" Static access method. """

if Singleton.\_\_instance == None:

Singleton()

print("here take your singleton inctance...\n")

return Singleton.\_\_instance

def \_\_init\_\_(self):

""" Virtually private constructor. """

if Singleton.\_\_instance != None:

raise Exception("This class is a singleton!")

else:

Singleton.\_\_instance = printer.adapter(operand.get\_polis())

printer.show\_beautifully(Singleton.\_\_instance)

o = Singleton()

s = Singleton.getInstance()

print(s, "\n")

def try\_to\_create():

try:

print("trying to create singleton instance once again...")

t = Singleton()

except Exception:

print('we got Exception("This class is a singleton!")')

print("Not able to create singleton instance twice. ")

print("Trying to get singleton instance...")

t = Singleton.getInstance()

print(t, "\n")

print("We got singleton instance!")

try\_to\_create()

Fasad

import pymysql.cursors

print\_db\_row=True

type\_test=False

def connect():

connection = pymysql.connect(

host = "localhost",

user = "root",

password = "qwertyui",

db = "sys\_analis",

charset = "utf8mb4",

cursorclass=pymysql.cursors.DictCursor

)

print("connected successfully!!!")

return connection

class printer():

def show\_beautifully(array):

print()

for x in array:

for y in x:

if(type(y)==type("test") and len(y)<9):

print(y, "\t\t", end=" ")

elif(type(y)==type("test") and len(y)<17):

print(y, "\t\t", end=" ")

elif(type(y)==type("test") and len(y)>17):

print(y, "\t", end=" ")

elif(type(y)==type(type\_test)):

print(y, "\t\t", end=" ")

else:

print(y, "\t\t", end=" ")

print("\n", end=" ")

def adapter(dbarray):

output=[]

a=[]

for b in dbarray[0]:

a.append(b)

output.append(a)

for row in dbarray:

elements=[]

for element in row:

elements.append(row[element])

output.append(elements)

return output

class operand():

def get\_consultant():

connection = connect()

try:

with connection.cursor() as cursor:

sql = "select \* from consultant"

cursor.execute(sql)

print()

ret=[]

for row in cursor:

if (print\_db\_row==True):

print(row)

ret.append(row)

finally:

connection.close()

return ret

def get\_opperation():

connection = connect()

try:

with connection.cursor() as cursor:

sql = "select \* from operation"

cursor.execute(sql)

print()

ret=[]

for row in cursor:

if (print\_db\_row==True):

print(row)

ret.append(row)

finally:

connection.close()

return ret

def get\_polis():

connection = connect()

try:

with connection.cursor() as cursor:

sql = "select \* from polis"

cursor.execute(sql)

print()

ret=[]

for row in cursor:

if (print\_db\_row==True):

print(row)

ret.append(row)

finally:

connection.close()

global type\_test

type\_test=ret[0]["name"]

return ret

class main():

def get\_table(table\_name):

chose={"consultant" : operand.get\_consultant,

"opperation" : operand.get\_opperation,

"polis" : operand.get\_polis}

printer.show\_beautifully(printer.adapter(chose[table\_name]()))

print("available tables: \n consultant\n polis\n operation\nchose needed one:")

main.get\_table('consultant')

State

import pymysql.cursors

import os

import keyboard

print\_db\_row=True

type\_test=False

def connect():

connection = pymysql.connect(

host = "localhost",

user = "root",

password = "qwertyui",

db = "sys\_analis",

charset = "utf8mb4",

cursorclass=pymysql.cursors.DictCursor

)

print("connected successfully!!!")

return connection

#######################################additional module##################

def cls():

os.system(['clear','cls'][os.name == 'nt'])

cls()

stop=False

def stop():

global stop

stop=True

keyboard.add\_hotkey('esc', stop)

def create\_main\_menu(menu):

global show\_menu

for i in range(0, len(menu)):

show\_menu.append(" ")

menu[i]+="\n"

show\_menu[i+1]+=menu[i]

cls()

for e in show\_menu:

print(e, end="")

def upchose():

global show\_menu

global current

global menu

if (current>0):

show\_menu[current]=show\_menu[current-1]

show\_menu[current-1]=">"

current-=1

cls()

print()

for e in show\_menu:

print(e, end="")

def downchose():

global show\_menu

global current

global menu

if (current<len(menu)-1):

show\_menu[current]=show\_menu[current+1]

show\_menu[current+1]=">"

current+=1

cls()

print()

for e in show\_menu:

print(e, end="")

def enter():

global show\_menu

global current

cls()

print()

for e in show\_menu:

print(e, end="")

chose(show\_menu[current+1])

menu=["consultant", "polis", "operation"]

show\_menu=[">"]

current=0

output=["\n", " "]

create\_main\_menu(menu)

def chose(point):

chose={" consultant\n" : operand1.get\_consultant,

" polis\n" : operand2.get\_polis,

" operation\n" : operand3.get\_operation

}

printer.show\_beautifully(main.adapter(chose[point]()))

chose[point]

#######################################################################

class printer():

def show\_beautifully(array):

for x in array:

for y in x:

if(type(y)==type("test") and len(y)<9):

print(y, "\t\t\t", end=" ")

elif(type(y)==type("test") and len(y)<17):

print(y, "\t\t", end=" ")

elif(type(y)==type("test") and len(y)>17):

print(y, "\t", end=" ")

elif(type(y)==type(type\_test)):

print(y, "\t\t", end=" ")

else:

print(y, "\t\t\t", end=" ")

print("\n", end=" ")

class operand1():

def get\_consultant():

connection = connect()

try:

with connection.cursor() as cursor:

sql = "select \* from consultant"

cursor.execute(sql)

print()

ret=[]

for row in cursor:

if (print\_db\_row==True):

print(row)

ret.append(row)

finally:

connection.close()

return ret

class operand2():

def get\_polis():

connection = connect()

try:

with connection.cursor() as cursor:

sql = "select \* from polis"

cursor.execute(sql)

print()

ret=[]

for row in cursor:

if (print\_db\_row==True):

print(row)

ret.append(row)

finally:

connection.close()

global type\_test

type\_test=ret[0]["name"]

return ret

class operand3():

def get\_operation():

connection = connect()

try:

with connection.cursor() as cursor:

sql = "select \* from operation"

cursor.execute(sql)

print()

ret=[]

for row in cursor:

if (print\_db\_row==True):

print(row)

ret.append(row)

finally:

connection.close()

return ret

class main():

def adapter(dbarray):

output=[]

a=[]

for b in dbarray[0]:

a.append(b)

output.append(a)

for row in dbarray:

elements=[]

for element in row:

elements.append(row[element])

output.append(elements)

return output

keyboard.add\_hotkey('up', upchose)

keyboard.add\_hotkey('down', downchose)

keyboard.add\_hotkey('enter', enter)

while True:

if (stop==True):

break

Strategy

import pymysql.cursors

import sys

import types

def connect():

connection = pymysql.connect(

host = "localhost",

user = "root",

password = "qwertyui",

db = "sys\_analis",

charset = "utf8mb4",

cursorclass=pymysql.cursors.DictCursor

)

print("connected successfully!!!")

return connection

connection = connect()

with connection.cursor() as cursor:

# SQL

sql = "SELECT \* FROM polis "

cursor.execute(sql)

if sys.version\_info[0] > 2:

create\_bound\_method = types.MethodType

else:

def create\_bound\_method(func, obj):

return types.MethodType(func, obj, obj.\_\_class\_\_)

class StrategyExample:

def \_\_init\_\_(self, func=None):

self.name = "Strategy Default"

if func:

self.execute = create\_bound\_method(func, self)

def execute(self):

print(self.name)

print()

def executeReplacement1(self):

print(self.name + " з випадку 1")

with connection.cursor() as cursor:

# SQL

sql = "SELECT \* FROM consultant "

cursor.execute(sql)

records = cursor.fetchmany(2)

print()

print("Total number of rows - ", cursor.rowcount)

for row in records:

print(row["name"], row["level"])

def executeReplacement2(self):

print(self.name + " з випадку 2")

with connection.cursor() as cursor:

# SQL

sql = "SELECT \* FROM polis"

cursor.execute(sql)

records = cursor.fetchmany(2)

print()

print("Total number of rows - ", cursor.rowcount)

for row in cursor:

print(row["duration"], row["amount"])

if \_\_name\_\_ == "\_\_main\_\_":

strat0 = StrategyExample()

strat1 = StrategyExample(executeReplacement1)

strat1.name = "Виведення лише імя та рівня консультанта:"

strat2 = StrategyExample(executeReplacement2)

strat2.name = "Виведення лише терміну та суми полісу:"

strat0.execute()

strat1.execute()

strat2.execute()

connection.close()

**Висновок:** на даній лабораторній роботі, я навчився створювати базу даних за діаграмами, будувати патерни для баз даних та підключати їх до бази даних