

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

# Кафедра програмного забезпечення та комп’ютерних систем

**Лабораторна робота №2**

з дисципліни

**«Бази даних і засоби управління»**

Виконала: студентка ІII курсу

ФПМ групи КП-72

Завгородня Анна

Перевірив(ла):

Київ – 2019

**Ознайомлення з базовими операціями СУБД PostgreSQL**

*Мета роботи:* здобуття практичних навичок проектування та побудови реляційних баз даних та створення прикладних програм з базами даних

*Завдання* роботи полягає у наступному:

1. Виконати нормалізацію бази даних, яка була створена у лабораторній роботі №1, до третьої нормальної форми (3НФ);
2. Реалізувати функціональні вимоги, наведені нижче.

*Функціональні вимоги:*

1. Реалізувати внесення, редагування та вилучення даних у базі засобами консольного інтерфейсу;
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі;
3. Забезпечити реалізацію пошуку за двома-трьома атрибутами з двох сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як перелічення, для логічного типу – значення True/False, для дат – у рамках діапазону дат;
4. Забезпечити реалізацію повнотекстового пошуку за будь-яким текстовим атрибутом бази даних засобами PostgreSQL з виділенням знайденого фрагменту.

*Вимоги до інтерфейсу користувача:*

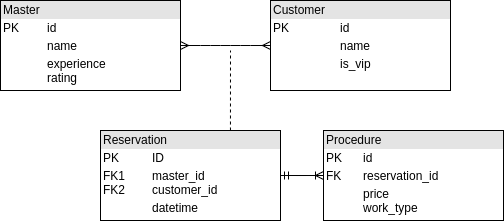
1. Використовувати консольний інтерфейс користувача.

**Варіант 8**

Пошук за атрибутами має відбуватися по переліченню строк та логічному типу.

Повнотекстовий пошук з обов’язковим входженням цілого слова або для випадку, коли вказане слово не входить в документ.

**Нормалізована модель даних**



Таблиця Master знаходиться у 1 нормальній формі , бо всі поля мають тільки атомарні значення. Таблиця Master знаходиться у 2НФ, бо знаходиться у 1НФ та кожний її неключовий атрибут функціонально повністю залежить від первинного ключа Id. Таблиця Master знаходиться у 3НФ , бо знаходиться у 2НФ та кожний її атрибут нетранзитивно залежить від первинного ключа.

Таблиця Customer знаходиться у 1 нормальній формі , бо всі поля мають тільки атомарні значення. Таблиця Customer знаходиться у 2НФ, бо знаходиться у 1НФ та кожний її неключовий атрибут функціонально повністю залежить від первинного ключа Id. Таблиця Customer знаходиться у 3НФ , бо знаходиться у 2НФ та кожний її атрибут нетранзитивно залежить від первинного ключа.

Таблиця Reservation знаходиться у 1 нормальній формі , бо всі поля мають тільки атомарні значення. Таблиця Reservation знаходиться у 2 НФ, бо знаходиться у 1НФ та кожний її неключовий атрибут функціонально повністю залежить від первинного ключа Id. Таблиця Reservation знаходиться у 3НФ , бо знаходиться у 2НФ та кожний її атрибут нетранзитивно залежить від первинного ключа.

Таблиця Procedure знаходиться у 1 нормальній формі , бо всі поля мають тільки атомарні значення. Таблиця Procedure знаходиться у 2 НФ, бо знаходиться у 1НФ та кожний її неключовий атрибут функціонально повністю залежить від первинного ключа Id. Таблиця Procedure знаходиться у 3НФ , бо знаходиться у 2НФ та кожний її атрибут нетранзитивно залежить від первинного ключа.

**Опис програми**

Програма створена за патерном MVC (Model-View-Controller). Складається відповідно з модулів model , view та controller.

У модулі model реалізовані функції , що здійснюють SQL запити до Бази Даних.

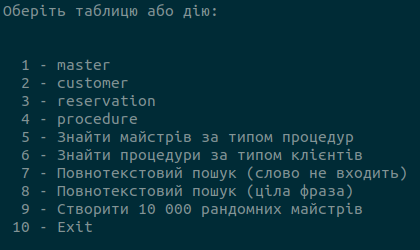
У модулі view реалізовані функції виводу даних з таблиць.

У класі Controller реалізовані функції для відповідних меню та допоміжні функції.

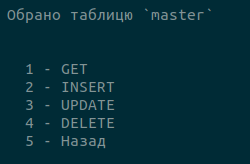
**Опис структури меню програми**

Меню програми можна розглядати як її концептуальну модель

Початкове меню:

****

Пункти 1-4:

****

Пункти 5-9: відображення результатів відповідного пошуку.

Пункт 10: вихід з програми.



**Лістинг програми**

*import* controller  
  
*if* \_\_name\_\_ == '\_\_main\_\_':  
 controller.Controller().show\_start\_menu()

**Controller**

*from* consolemenu *import* SelectionMenu  
  
*import* model  
*import* view  
*import* scanner  
  
  
*class* Controller:  
 *def \_\_init\_\_*(*self*):  
 *self*.model = model.Model()  
 *self*.view = view.View()  
  
 *self*.tables = *list*(model.TABLES.keys())  
  
 *def* get\_table\_name(*self*, index):  
 *try*:  
 *return self*.tables[index]  
 *except IndexError*:  
 *return None*  
  
 *def* show\_start\_menu(*self*, subtitle='', \*\*kwargs):  
 menu\_options = *self*.tables + ['Знайти майстрів за типом процедур',  
 'Знайти процедури за типом клієнтів',  
 'Повнотекстовий пошук (слово не входить)',  
 'Повнотекстовий пошук (ціла фраза)',  
 'Створити 10 000 рандомних майстрів']  
 next\_steps = [*self*.show\_table\_menu] \* *len*(*self*.tables) + [  
 *self*.get\_masters\_by\_procedures,  
 *self*.get\_procedure\_by\_client\_type,  
 *self*.fts\_without\_word,  
 *self*.fts\_phrase,  
 *self*.create\_random\_masters  
 ]  
 menu = SelectionMenu(menu\_options, subtitle=subtitle,  
 title="Оберіть таблицю або дію:")  
 menu.show()  
  
 index = menu.selected\_option  
 *if* index < *len*(menu\_options):  
 table\_name = *self*.get\_table\_name(index)  
 next\_step = next\_steps[index]  
 *try*:  
 next\_step(table\_name=table\_name)  
 *except Exception as* err:  
 *self*.show\_start\_menu(subtitle=*str*(err))  
 *else*:  
 *print*('Пака!!!!')  
  
 *def* show\_table\_menu(*self*, table\_name, subtitle=''):  
 next\_steps = [*self*.get, *self*.insert, *self*.update, *self*.delete, *self*.show\_start\_menu]  
 menu = SelectionMenu(  
 ['GET', 'INSERT', 'UPDATE', 'DELETE'], subtitle=subtitle,  
 title=f'Обрано таблицю `{table\_name}`', exit\_option\_text='Назад', )  
 menu.show()  
  
 next\_step = next\_steps[menu.selected\_option]  
 next\_step(table\_name=table\_name)  
  
 *def* get(*self*, table\_name):  
 filter\_by = scanner.input\_dict(table\_name, 'За чим фільтрувати запит? Залиште пустим щоб отримати всі рядки:')  
 data = *self*.model.get(table\_name, \*\*filter\_by)  
 *self*.view.print\_entities(table\_name, data)  
 scanner.press\_enter()  
 *self*.show\_table\_menu(table\_name)  
  
 *def* insert(*self*, table\_name):  
 new\_values = scanner.input\_dict(table\_name, 'Введіть нові значення:')  
 *self*.model.insert(table\_name, \*\*new\_values)  
 *self*.show\_table\_menu(table\_name, 'Вставка відбулася успішно')  
  
 *def* update(*self*, table\_name):  
 filter\_by = scanner.input\_dict(table\_name, 'Який рядок треба змінити?:', limit=1)  
 new\_values = scanner.input\_dict(table\_name, 'Введіть нові значення:')  
 *self*.model.update(table\_name, *list*(filter\_by.items())[0], \*\*new\_values)  
 *self*.show\_table\_menu(table\_name, 'Оновлення відбулося успішно')  
  
 *def* delete(*self*, table\_name):  
 filter\_by = scanner.input\_dict(table\_name, 'Які рядки треба видалити?')  
 *self*.model.delete(table\_name, \*\*filter\_by)  
 *self*.show\_table\_menu(table\_name, 'Видалення відбулося успішно')  
  
 *def* get\_masters\_by\_procedures(*self*, \*\*kwargs):  
 procedures = scanner.input\_simple('Через кому введіть процедури:').split(',')  
 procedures = [p.strip() *for* p *in* procedures]  
 data = *self*.model.get\_masters\_by\_procedures(procedures)  
 *self*.view.print\_entities(f'Майстри які робили процедури={procedures}', data)  
 scanner.press\_enter()  
 *self*.show\_start\_menu()  
  
 *def* get\_procedure\_by\_client\_type(*self*, \*\*kwargs):  
 is\_vip = scanner.input\_simple('Чи є клієнт ВІПом?').lower() *in* ('t', 'true', '+', 'yes', 'y')  
 data = *self*.model.get\_procedure\_by\_client\_type(is\_vip)  
 *self*.view.print\_entities(  
 f'Майстри які обслуговували {"" *if* is\_vip *else* "не "}ВІП клієнтів', data)  
 scanner.press\_enter()  
 *self*.show\_start\_menu()  
  
 *def* fts\_without\_word(*self*, \*\*kwargs):  
 word = scanner.input\_simple('Яке слово має бути відсутнє у документі?')  
 data = *self*.model.fts\_without\_word(word)  
 *self*.view.print\_entities(f'Документи, в яких немає слова `{word}`', data)  
 scanner.press\_enter()  
 *self*.show\_start\_menu()  
  
 *def* fts\_phrase(*self*, \*\*kwargs):  
 phrase = scanner.input\_simple('Введіть фразу, за якою хочете здійснити повнотекстовий пошук:')  
 data = *self*.model.fts\_phrase(phrase)  
 *self*.view.print\_entities(f'Документи, в яких є фраза `{phrase}`', data)  
 scanner.press\_enter()  
 *self*.show\_start\_menu()  
  
 *def* create\_random\_masters(*self*, \*\*kwargs):  
 *self*.model.create\_random\_masters()  
 *self*.show\_start\_menu('10 000 випадкових майстрів додано')

**View**

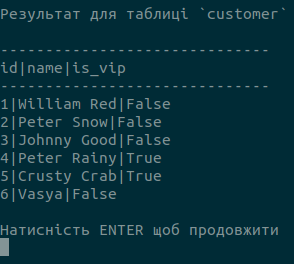
*class* View:  
 *def \_\_init\_\_*(*self*):  
 *self*.SEPARATOR\_WIDTH = 30  
  
 *def* print\_entities(*self*, table\_name, data):  
 entities, cols = data  
 separator\_line = '-' \* *self*.SEPARATOR\_WIDTH  
  
 *print*(f'Результат для таблиці `{table\_name}`', end='\n\n')  
 *print*(separator\_line)  
 *print*('|'.join(cols))  
 *print*(separator\_line)  
 *for* entity *in* entities:  
 *print*('|'.join([*str*(col) *for* col *in* entity]))

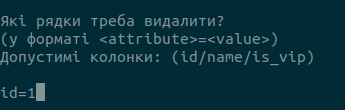
**Model**

*import* psycopg2  
  
TABLES = {  
 'master': ('id', 'name', 'experience', 'rating'),  
 'customer': ('id', 'name', 'is\_vip'),  
 'reservation': ('id', 'master\_id', 'customer\_id', 'datetime'),  
 'procedure': ('id', 'reservation\_id', 'price', 'work\_type')  
}  
  
  
*class* Model:  
 *def \_\_init\_\_*(*self*):  
 *self*.conn = psycopg2.connect("dbname='kpi' user='admin'"  
 "host='localhost' password='admin'")  
 *self*.cursor = *self*.conn.cursor()  
  
 *def* create\_tables(*self*):  
 *with open*('scripts/create.sql') *as* file:  
 command = file.read()  
 *self*.cursor.execute(command)  
 *self*.conn.commit()  
  
 *def* get(*self*, table\_name, \*\*kwargs):  
 command = f'SELECT \* FROM {table\_name}'  
  
 *if* kwargs:  
 conditions = [f"{column}='{kwargs[column]}'" *for* column *in* kwargs]  
 command = f'{command} where {" and ".join(conditions)}'  
  
 *self*.cursor.execute(command)  
 *return self*.cursor.fetchall(), TABLES[table\_name]  
  
 *def* insert(*self*, table\_name, \*\*new\_values):  
 *if not* new\_values:  
 *raise Exception*('Не вказані поля, які треба заповнити')  
  
 *try*:  
 columns = new\_values.keys()  
 values = [f"'{value}'" *for* value *in* new\_values.values()]  
 command = f'INSERT INTO {table\_name} ({", ".join(columns)})' + \  
 f'VALUES ({", ".join(values)})'  
  
 *self*.cursor.execute(command)  
 *finally*:  
 *self*.conn.commit()  
  
 *def* update(*self*, table\_name, condition, \*\*new\_values):  
 *if not* new\_values:  
 *raise Exception*('Не вказані поля, які треба оновити')  
  
 *try*:  
 column, value = condition  
 updates = ', '.join([f"{column} = '{new\_values[column]}'" *for* column *in* new\_values])  
 command = f'UPDATE {table\_name} SET {updates} WHERE {column}={value}'  
  
 *self*.cursor.execute(command)  
 *finally*:  
 *self*.conn.commit()  
  
 *def* delete(*self*, table\_name, \*\*filter\_by):  
 *if not* filter\_by:  
 *raise Exception*('Не вказані умови для рядків, які треба видалити')  
  
 *try*:  
 conditions = [f"{column}='{filter\_by[column]}'" *for* column *in* filter\_by]  
 command = f'DELETE FROM {table\_name} WHERE {" and ".join(conditions)}'  
  
 *self*.cursor.execute(command)  
 *finally*:  
 *self*.conn.commit()  
  
 *def* get\_masters\_by\_procedures(*self*, procedures):  
 procedures = [f"'{p.lower()}'" *for* p *in* procedures]  
 command = f'''  
 SELECT name FROM master  
 JOIN reservation r on master.id = r.master\_id  
 JOIN procedure p on p.reservation\_id = r.id  
 WHERE lower(work\_type) in ({", ".join(procedures)})  
 GROUP BY name;'''  
  
 *self*.cursor.execute(command)  
 *return self*.cursor.fetchall(), ('name',)  
  
 *def* get\_procedure\_by\_client\_type(*self*, is\_vip):  
 command = f'''  
 SELECT work\_type FROM procedure  
 JOIN reservation r on "procedure".reservation\_id = r.id  
 JOIN customer c2 on r.customer\_id = c2.id  
 WHERE c2.is\_vip={is\_vip}  
 GROUP BY work\_type;'''  
  
 *self*.cursor.execute(command)  
 *return self*.cursor.fetchall(), ('work\_type',)  
  
 *def* fts\_without\_word(*self*, word):  
 *with open*('scripts/fts\_without\_word.sql') *as* file:  
 sql = file.read().replace('$WORD', word)  
 *self*.cursor.execute(sql)  
 *return self*.cursor.fetchall(), ('master\_name', 'customer\_name', 'work\_type')  
  
 *def* fts\_phrase(*self*, phrase):  
 *with open*('scripts/fts\_phrase.sql') *as* file:  
 sql = file.read().replace('$PHRASE', phrase)  
 *self*.cursor.execute(sql)  
 *return self*.cursor.fetchall(), ('master\_name', 'customer\_name', 'work\_type')  
  
 *def* create\_random\_masters(*self*):  
 *try*:  
 *with open*('scripts/random.sql') *as* file:  
 sql = file.read()  
 *self*.cursor.execute(sql)  
 *finally*:  
 *self*.conn.commit()

**Скріншот результатів виконання операції вилучення**

Спроба вилучення кортежу на який є посилання







Вдале видалення і деомонстрація результату вилучення

