НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

# Кафедра системного програмування та спеціалізованих комп’ютерних систем

**Лабораторна робота №2**

з дисципліни

**«Бази даних і засоби управління»**

Тема: «**Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL**»

Виконав: студент III курсу

ФПМ групи КВ-81

Савченко В. С.

Київ – 2020

*Метою роботи* є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

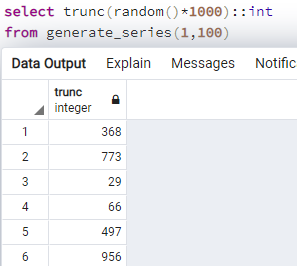
*Загальне завдання* роботи полягає у наступному:

1. Реалізувати функції внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

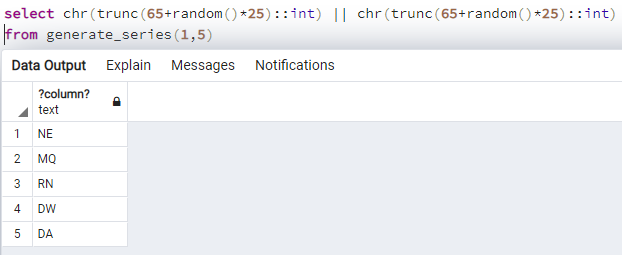
*Деталізоване завдання*:

1. Забезпечити можливість уведення/редагування/вилучення даних у таблицях бази даних з можливістю контролю відповідності типів даних атрибутів таблиць (рядків, чисел, дати/часу). Для контролю пропонується два варіанти: контроль при введенні (валідація даних) та перехоплення помилок (try..except) від сервера PostgreSQL при виконанні відповідної команди SQL. Особливу увагу варто звернути на дані таблиць, що мають зв’язок 1:N. При цьому з боку батьківської таблиці необхідно контролювати **вилучення** рядків за умови наявності даних у підлеглій таблиці. З точки зору підлеглої таблиці варто контролювати наявність відповідного рядка у батьківській таблиці при виконанні **внесення** нових даних. Унеможливити виведення програмою системних помилок на екрані шляхом їх перехоплення і адекватної обробки. Внесення даних виконується користувачем у консольному вікні програми.
2. Забезпечити можливість автоматичної генерації великої кількості даних у таблицях за допомогою вбудованих у PostgreSQL функцій роботи з псевдовипадковими числами. Дані мають бути згенерованими **не мовою програмування, а відповідним SQL-запитом**!

Приклад генерації 100 псевдовипадкових чисел:



Приклад генерації 5 псевдовипадкових рядків:



Приклад генерації псевдовипадкової мітки часу з діапазону [доступний за посиланням](https://stackoverflow.com/questions/22964272/postgresql-get-a-random-datetime-timestamp-between-two-datetime-timestamp/22965061).

Кількість даних для генерування має вводити користувач з клавіатури. Для тесту взяти 100 000 записів для однієї-двох таблиць.

Особливу увагу слід звернути на відповідність даних вимогам зовнішніх ключів з метою уникнення помилок порушення обмежень цілісності (foreign key).

1. Для реалізації пошуку необхідно підготувати 3 запити, що включають дані з декількох таблиць і фільтрують рядки за 3-4 атрибутами цих таблиць. Забезпечити можливість уведення конкретних значень констант для фільтрації з клавіатури користувачем. Крім того, після виведення даних необхідно вивести час виконання запиту у мілісекундах. Перевірити швидкодію роботи запитів на попередньо згенерованих даних.
2. Програмний код організувати згідно шаблону Model-View-Controller(MVC).  Приклад організації коду згідно шаблону доступний [за даним посиланням](https://www.giacomodebidda.com/mvc-pattern-in-python-introduction-and-basicmodel/). При цьому модель, подання та контролер мають бути реалізовані у окремих файлах. Для доступу до бази даних використовувати **лише мову SQL** (без ORM).

Рекомендована бібліотека взаємодії з PostgreSQL Psycopg2: http://initd.org/psycopg/docs/usage.html)

*Вимоги до інтерфейсу користувача*

Використовувати консольний інтерфейс користувача.

*Вимоги до інструментарію*

 Середовище для відлагодження SQL-запитів до бази даних – PgAdmin4.

 Мова програмування – Python 3.6-3.7

 Середовище розробки програмного забезпечення – PyCharm Community Edition 2020.

*Вимоги до оформлення звіту лабораторної роботи у електронному вигляді*

Опис (файл README.md) лабораторної роботи у  **репозиторії** **GitHub** включає: назву лабораторної роботи, структуру бази даних з лабораторної роботи №1.

Репозиторій має містить файл звіту у форматі PDF та програмний код файлів мовою Python (або іншою).

Звіт у форматі PDF має містити: титульний аркуш, завдання, URL репозиторію з вихідним кодом та відповіді на вимоги до звітування щодо пунктів 1-4 деталізованого завдання:

*Вимоги до пункту №1 деталізованого завдання:*

* ілюстрації обробки виняткових ситуацій (помилок) при уведенні/вилучення даних;
* ілюстрації валідації даних при уведенні користувачем.

*Вимоги до пункту №2 деталізованого завдання:*

* копії екрану (ілюстрації) з фрагментами згенерованих даних таблиць;
* копії SQL-запитів, що ілюструють генерацію при визначених вхідних параметрах.

*Вимоги до пункту №3 деталізованого завдання:*

* ілюстрації уведення пошукового запиту та результатів виконання запитів;
* копії SQL-запитів, що ілюструють пошук з зазначеними початковими параметрами

*Вимоги до пункту №4 деталізованого завдання:*

* ілюстрації програмного коду з репозиторію Git.

1.Реалізувати функції внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.

У даній роботі було використано базу даних предметої галузі «Бібліотека».

Концептуальна схема виглядає так

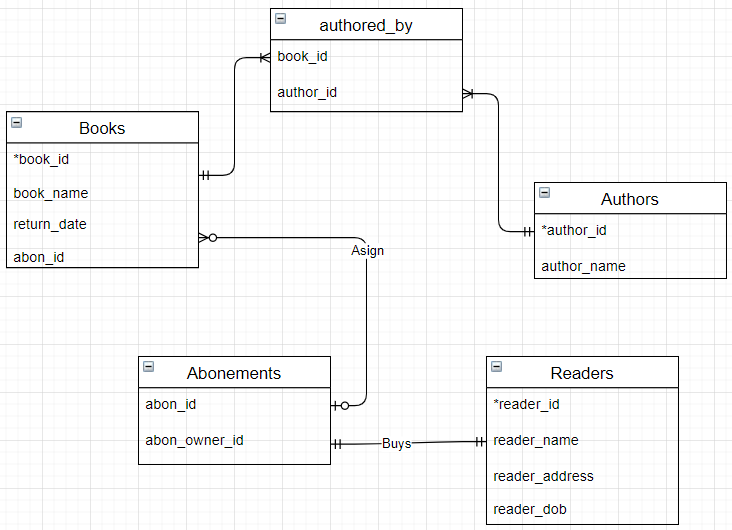
****

Рис. 1 концептуальна схема БД «Бібліотека»

Програму для виконання запитів методами консольного інтерфейсу було розроблено спочатку(для ознайомлення) на мові програмування Python і тільки після детального усвідомлення проблематики переведено у покращений та більш абстрагований алгоритм на мові програмування С#.

Наводити повний код програми у цьому документі не має змоги, бо це сильно збільшить обсяги звіту, тому завантажу обидві програми на github.

На прикладі таблиці «Читач» розберемо операції додавання, зчитування, оновлення та вилучення записів.

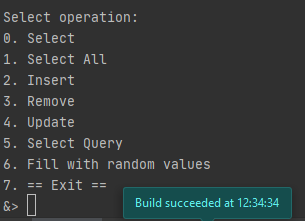
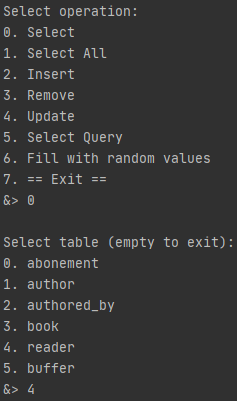
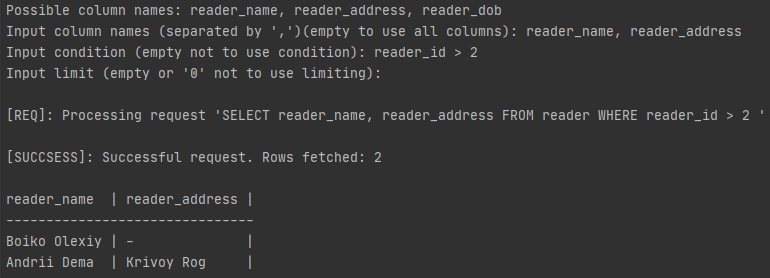


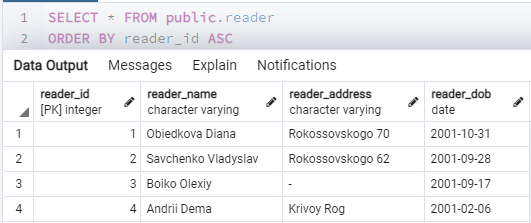
Рис. 1.2 головне меню програми.

У цьому меню обираємо пункт 0. Select і таблицю «Читач»



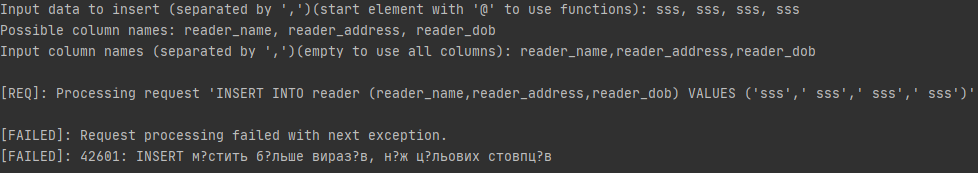
Далі програма пропонує нам ввести усі необхідні дані

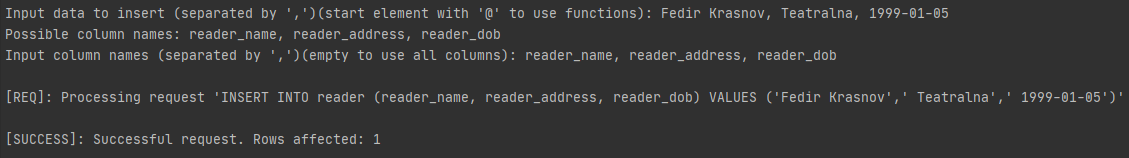


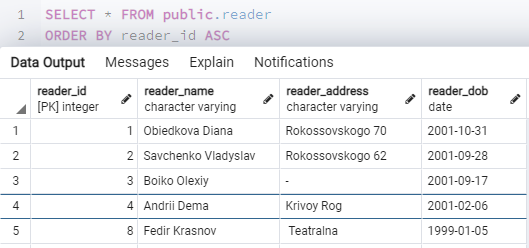
Як бачимо, запит було виконано, але питання у тому,правильності. Нижче наведено скріншот початкового стану таблиці

Отже, правильно.

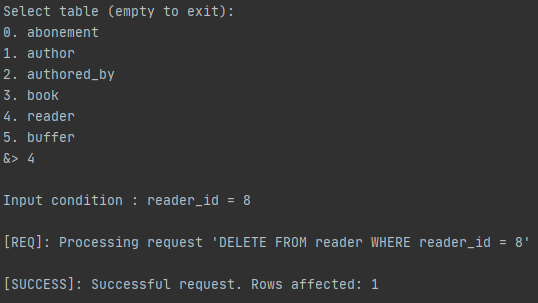
Тепер змінимо мету. Припустимо, що ми хочемо ввести некоректні дані у цю таблицю.



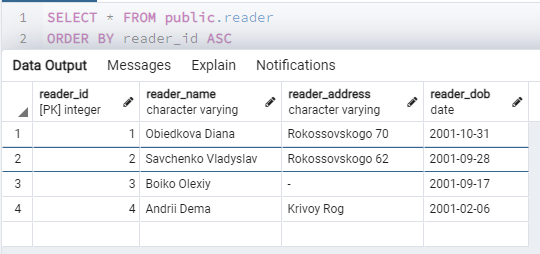
Як бачимо, програма коректно опрацьовує помилки у запитах. Тепер спробуємо додати вже валідний запис та потім вилучити його. Нехай додамо обліковий запис Краснова Федора. 

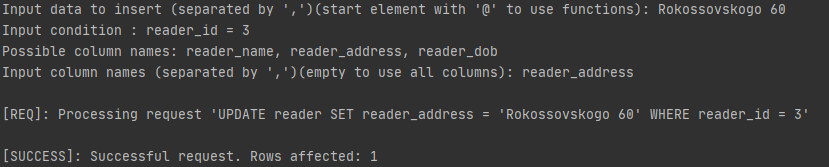
Тепер перевіримо таблицю. 

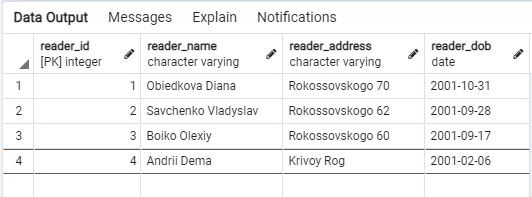
Як, бачимо, запис було додано. Тепер давайте видалимо цей же запис з таблиці.



Перевіряємо таблицю:

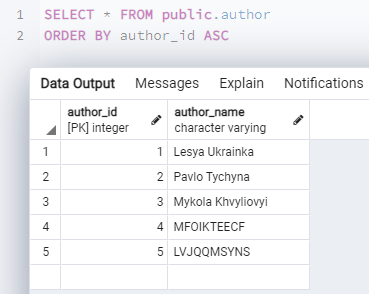


Бачимо, що все працює коректно. Тепер для повноти картини, змінимо reader\_address у Олексія. 

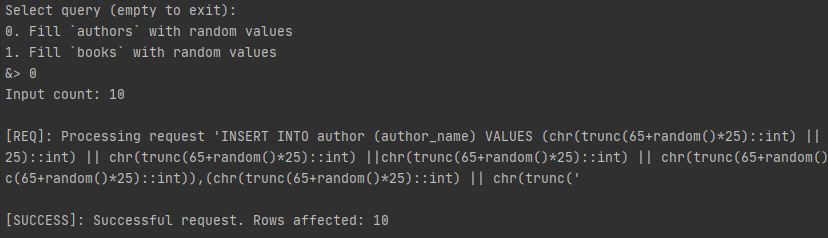
Знову ж таки перевіряємо таблицю

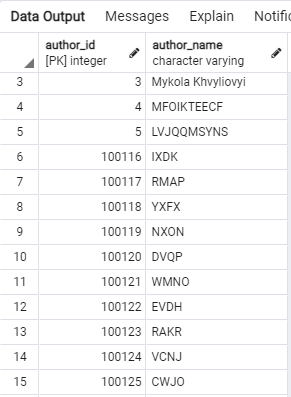
Все працює коректно.

2.Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.

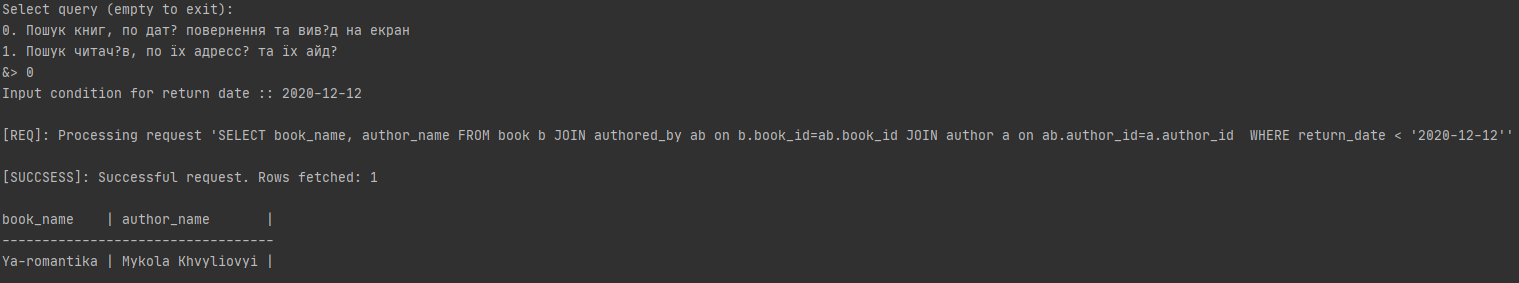
Тепер продемонструю пакетне генерування даних. Для цього спробуємо додати у таблицю «Автори» 10 елементів 

Таблиця «Автори» до виконання запиту. Тепер виконаємо генерування.



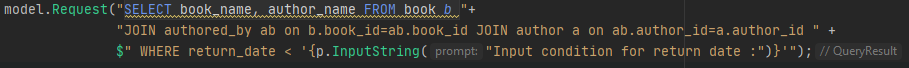
Тепер перевіримо саму таблицю 

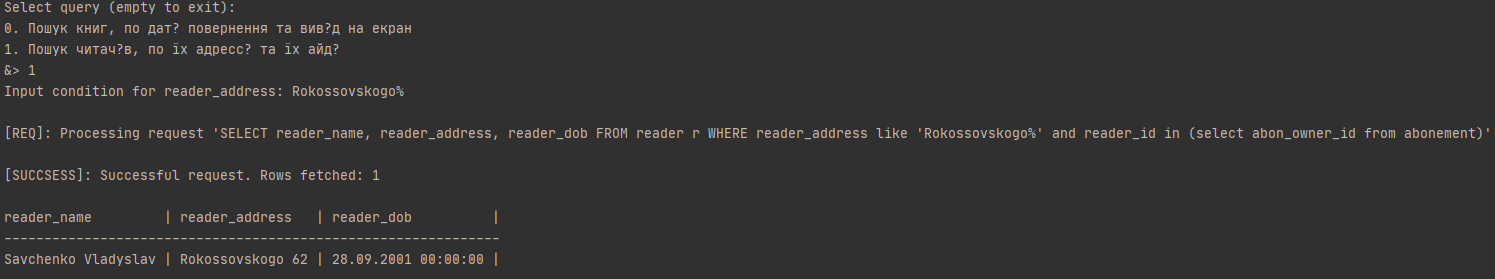
Отримали такі великі айді у наслідок того, що перед виконанням звіту були проведені декілька тестів.

3.Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.

1й запит та результат.

Сам запит наводжу нижче



2й запит і результат:

Копія запиту:

